Oracle® Fusion Middleware

Developer's Guide for Oracle Adaptive Access Manager

Release 11*g* (11.1.1)

E15480-06

August 2011

ORACLE®

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

E15480-06

Primary Author: Priscilla Lee

Contributors: Niranjan Ananthapadmanabha, Mandar Bhatkhande, Sree Chitturi, Josh Davis, Jordan Douglas, Philomina Dorai, Daniel Joyce, Mark Karlstrand, Wei Jie Lee, Srinivas Nagandla, Paresh Raote, Jatin Rastogi, Jim Redfield, Nandini Subramani, Elangovan Subramanian, Vidhya Subramanian, Dawn Tyler, Sachin Vanungare, Saphia Yunaeva, and Xiaobin Zheng.

Contents

3 Integrating Native .NET Applications

4 Integrating Native Java Applications

5 Native API for OTP Challenge

8 Customizing the OAAM Server

9 Customizing User Flow

10 Using Virtual Authentication Devices

11 Implementing OTP Anywhere

20 Developing Custom Challenge Processors

21 Creating a View of a Non-OAAM Database

22 Developing a Custom Loader for OAAM Offline

Part VII Troubleshooting

23 FAQ/Troubleshooting

Preface

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

The Preface covers the following topics:

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

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Related Documents

For more information, see the following documents in the Oracle Fusion Middleware 11*g* 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 with Oracle Security Token Service*
- *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.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Introduction to the Developer's Guide

Oracle Adaptive Access Manager provides a variety of mechanisms for integrating with custom applications and custom development.

The *Oracle Fusion Middleware Developer's Guide for Oracle Adaptive Access Manager* provides information to help developers integrate and customize Oracle Adaptive Access Manager, migrate 10*g* native applications, 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 Proxy
- Part III - Customization and extensions
- Part IV - Oracle Adaptive Access Manager, Oracle Access Manager, and Oracle Identity Manager integration
- Part V - Migration and lifecycle management
- Part VI - Custom development
- 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

Applications can integrate natively with Oracle Adaptive Access Manager using APIs. 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."

OTP Integration

Oracle Adaptive Access Manager's Native OTP API offers a way to add another factor to a traditional user name/password authentication scheme.

For information on OTP integration, see Chapter 5, "Native API for OTP Challenge."

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 chapter:

- Chapter 6, "Oracle Adaptive Access Manager Proxy"

1.3 Customizations and Extensions

Part III provides instructions and reference material for the following customizing and extending features of Oracle Adaptive Access Manager:

Customizing Oracle Adaptive Access Manager

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

For information on using the extensions shared library for customization of Oracle Adaptive Access Manager, see Chapter 7, "Customizing Oracle Adaptive Access Manager."

Customizing the OAAM Server

The user interface provided by the OAAM Server Web application can be easily customized to achieve the look and feel of the customer applications. You can configure OAAM Server to support one or more Web application authentication and user registration flows.

For information on the customization of OAAM Server, see Chapter 8, "Customizing the OAAM Server."

Customizing User Flow

OAAM supports the customization of user flow. For information, refer to Chapter 9, "Customizing User Flow."

Virtual Authentication Devices

Oracle Adaptive Access Manager includes unique functionality to protect end users while interacting with a protected web application. The virtual authentication devices hardens the process of entering and transmitting authentication credentials and provide end users with verification they are authenticating on the valid application.

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

For information on the customization of virtual authentication devices, see Chapter 10, "Using Virtual Authentication Devices."

One-Time Password

Oracle Adaptive Access Manager 11*g* provides the framework to support One Time Password (OTP) authentication with Oracle User Messaging Service (UMS) as a method of delivery out of the box.

For instructions to configure OTP to leverage UMS as a method of delivery, refer to Chapter 11, "Implementing OTP Anywhere."

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 12, "Configurable Actions" describes how to integrate a Configurable Action with the Oracle Adaptive Access Manager software.

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 13, "Device Registration."

Device Identification

For most typical deployments, the out-of-the-box device identification satisfies client requirements. Out-of-the-box Device Identification uses data from browser and OAAM flash movie. The following are the typical scenarios when you could consider extending device identification:

- The OAAM flash movie cannot be used to obtain client details as the client side browser does not support Flash (example: iPhone, iPad, and so on)
- There is a need to extract stronger device identification data from the client using a non-flash plug-in that can run inside the browser

For information on how to extend device identification in a typical deployment refer to Chapter 14, "Extending Device Identification."

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 15, "Flash Fingerprinting."

1.4 Authentication and Password Management Integration

Benefits of the Oracle Access Manager-Oracle Adaptive Access Manager-Oracle Identity Manager integration is presented in Chapter 16, "Access and Password Management Integration."

1.5 Migration and Lifecycle Management

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

For the steps involved in migrating an existing natively integrated 10.1.4.5 application that is currently using SOAP authentication to 11*g*, refer to Chapter 17, "Migrating Native Applications to OAAM 11g.".

For examples for handling lifecycle configuration changes, refer to Chapter 18, "Handling Lifecycle Management Changes."

1.6 Custom Development

Custom development instructions are in the following chapters:

- Chapter 19, "Creating OAAM Oracle BI Publisher Reports"
- Chapter 20, "Developing Custom Challenge Processors"
- Chapter 21, "Creating a View of a Non-OAAM Database"
- Chapter 22, "Developing a Custom Loader for OAAM Offline"

1.7 Troubleshooting/FAQ

Chapter 23, "FAQ/Troubleshooting" provides troubleshooting tips and answers to frequently asked questions.

Part I

Native Integration

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"
- Chapter 5, "Native API for OTP Challenge"

2

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.

Native Oracle Adaptive Access Manager integration involves customizing your application to include OAAM API calls at various stages of the login process. 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

Example Application

The example application is available as a form of documentation to illustrate how to call the product APIs. It is not intended as production code. For example, the sample application does not have proper error handling; it only provides basic elements of API usage. Customers implementing a native integration should develop their own application using the sample application as a reference only.

Note: Custom applications developed for these deployments are not supported directly. However, Oracle Support Services can assist customers with product issues, such as if customers encounter problems when using the provided APIs.

2.1 Overview

To integrate with Oracle Adaptive Access Manager, the application can use the native API. Choose one of the following native API options:

- 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.

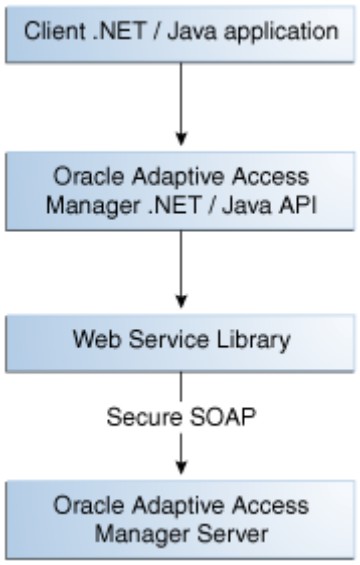
The SOAP service wrapper API enables you to create SOAP objects and invoke SOAP calls and abstracts the SOAP Web Service Definition Language (WSDL) and other Web services details from the application code. Libraries for this API are available for the following languages: Java, .NET, and C++.

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

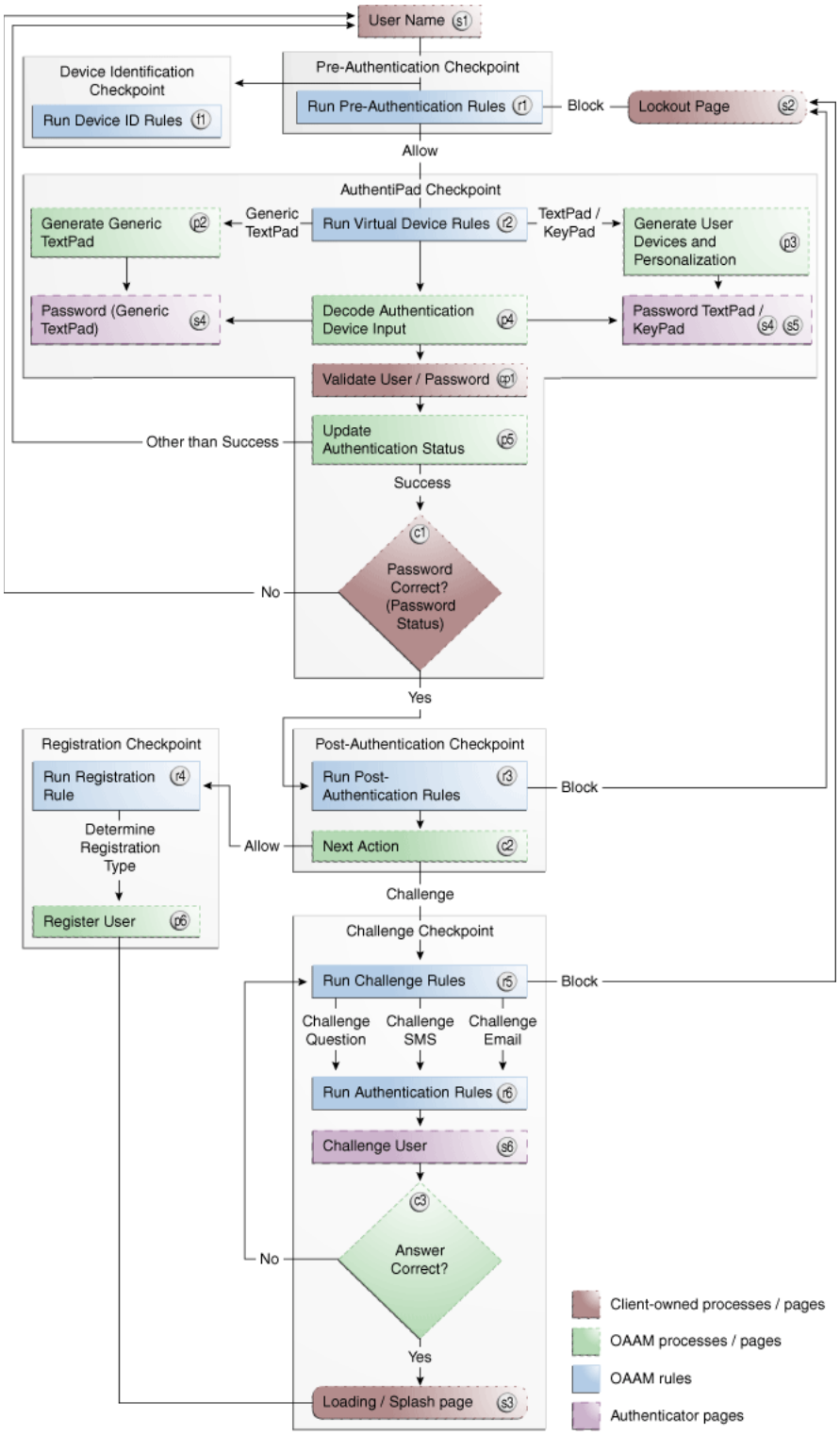
This integration consolidates virtual authentication devices and knowledge-based authentication. Globalized virtual authentication device image files including registration flows must be developed by the deployment team.

Figure 2–2 illustrates an authentication flow example that uses these three solutions (virtual authentication devices, knowledge-based authentication, One-Time Password). Note that the flow illustrated is an example and that other authentication flows are possible.

The details of the stages in the Figure 2–2 are explained in the following sections:

- User Name Page (S1)
- Device Fingerprint Flow (F1)
- Run Pre-Authentication Rules (R1)
- Run Virtual Authentication Device Rules (R2)
- Decode Virtual Authentication Device Input (P4)
- Validate User and Password (CP1)
- Run Post-Authentication Rules (R3)
- Check Registration for User (C2)
- Run Registration Required Rules (R4)
- Enter Registration Flow (P6)
- Run Challenge Rules (R5)
- Run Authentication Rules (R6)
- Challenge the User (S6)
- Check Answers to Challenge (C3)
- Lock Out Page (S2)
- Landing or Splash Page (S3)

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



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

2.2.1.2 Device Fingerprint Flow (F1)

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() APIs that construct the fingerprint are: ■ VCryptServletUtil.getBrowserFingerPrint(userAgent, language, country, variant); ■ VCryptServletUtil.getFlashFingerPrint(client, fpStr);	For method details on updateLog(), see Section 4.5.6, "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.jps page
Oracle Adaptive Access Manager Sample	handleFlash.jsp	Sets the flashCookie if the browser is flash-enabled

Cookies in Device Identification

Oracle Adaptive Access Manager uses two types of cookies to perform device identification.

One is the browser cookie (also known as secure cookie) and the other is the flash cookie (also known as digital cookie).

The browser cookie value is constructed using the browser user agent string. The flash cookie value is constructed using data from the OAAM flash movie.

The following is sample code to fingerprint the device using browser and flash cookies. Refer to code in `handleFlash.jsp` for details:

```
//Get Browse/Secure cookie
String secureCookie = getCookie(request, "bharosa");
Locale locale = request.getLocale();
String browserFp =
VCryptServletUtil.getBrowserFingerPrint(request.getHeader("user-agent"),
locale.getLanguage(),
locale.getCountry(), locale.getVariant());



String client = request.getParameter("client");
String fpStr = request.getParameter("fp");
String flashFp = bharosaHelper.constructFlashFingerPrint( client, fpStr );

//Get the flash cookie
String flashCookie = request.getParameter("v");
CookieSet cookieSet = bharosaHelper.fingerPrintFlash(bharosaSession,
bharosaSession.getRemoteIPAddr(), request.getRemoteHost(),
```

```
BharosaEnumAuthStatus.PENDING, secureCookie, browserFp, flashCookie, flashFp);
```

2.2.1.3 Run 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 4.6.1, "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 Run 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 inTable 2–3.

Table 2–3 Virtual Authentication Device Rules APIs

Module	APIs	Description
Server	VCryptRulesEngine::processRules()	For method details, see Section 4.6.1, "processRules."
Oracle Adaptive Access Manager Sample	password.jsp	Invokes rules to identify the user's virtual authentication device type Creates the virtual authentication device, names it, and sets all initial background frames Invokes kbimage.jsp as configured Forwards to page handlePassword.jsp
BharosaHelper	BharosaHelper::getAuthentiPad()	

2.2.1.4.1 Generate 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 Generic TextPad APIs

Module	APIs	Description
Server	VCryptAuth::getUserByLoginId() You can obtain an instance of VCryptAuth by calling VCryptAuthUtil.getVCryptAuthInstance().	For method details, see Section 4.5.7, "getUserByLoginId."
Oracle Adaptive Access Manager Sample	Password.jsp	Invokes rules to identify the virtual authentication device type to use; the default is KeyPad Creates the virtual authentication device, names it, and sets all initial background frames Invokes kbimage.jsp as configured Forwards to page handlePassword.jsp
BharosaHelper	BharosaHelper:: createPersonalizedAuthentiPad () BharosaHelper::createAuthentiPad()	
Client	AuthentiPad::getHTML()	

2.2.1.4.2 Generate a Personalized TextPad or KeyPad (P3) A personalized TextPad is used for users who have registered with Oracle Adaptive Access Manager. Figure 2–5 andFigure 2–6 illustrate personalized text and key virtual authentication devices.

Figure 2–5 Personalized TextPad

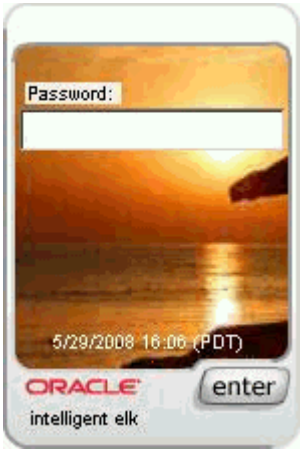


Figure 2–6 Personalized KeyPad

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 4.5.7, "getUserByLoginId."
Oracle Adaptive Access Manager Sample	password.jsp	Invokes rules to identify the virtual authentication device type to use; the default is KeyPad
		Creates the virtual authentication device, names it, and sets all initial background frames
		Forwards to page handlePassword.jsp
		Invokes kbimage.jsp as configured
BharosaHelper	BharosaHelper:: createPersonalizedAuthentiPad ()	
	BharosaHelper::createAuthentiPad()	
Client	AuthentiPad::getHTML()	

2.2.1.4.3 Display 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 `<img>` 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	Invokes rules to identify the virtual authentication device type to use; the default is KeyPad
		Creates the virtual authentication device, names it, and sets all initial background frames
		Invokes kbimage.jsp as configured
		Forwards to page handlePassword.jsp
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()	

2.2.1.5 Decode 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 serialized and stored in the database or the file system.

The virtual authentication device is stored in session because it is used to decode the input. This is needed for virtual authentication devices like PinPad and KeyPad where the user input is not clear text. For consistency it is performed for all virtual authentication devices since they are designed to be able to be used interchangeably.

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	

2.2.1.6 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 OAAM 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

2.2.1.6.1 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 4.5.9, "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()	

2.2.1.6.2 Password Status (C1) Depending on the password authentication status, the user is directed to the retry page or to post-authentication.

2.2.1.7 Run 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.

The APIs used for post-authentication are listed in Table 2–10.

Table 2–10 Post-Authentication Rules Reference APIs

Module	APIs	Description
Server	VCryptRulesEngine::processRules()	For method details, see Section 4.6.1, "processRules."
Oracle Adaptive Access Manager Sample	handlePassword.jsp	Calls BharosaHelper::runPostAuthRules which returns: ALLOW BLOCK CHALLENGE If ALLOW: BharosaHelper::runRegistrationRules returns ALLOW REGISTER_QUESTIONS REGISTER_USER_INFO REGISTER_USER SYSTEM_ERROR If CHALLENGE: forward_challengePage
BharosaHelper	BharosaHelper::runPostAuthRules()	

2.2.1.8 Check Registration for User (C2)

Rules are run to check registration; if the user is not registered, he is directed to do so.

2.2.1.9 Run 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 4.6.1, "processRules."
Oracle Adaptive Access Manager Sample	password.jsp	Invokes rules to identify the virtual authentication device type to use; the default is KeyPad
		Creates the virtual authentication device, names it, and sets all initial background frames
		Invokes kbimage.jsp as configured
		Forwards to page handlePassword.jsp
BharosaHelper	BharosaHelper::getAuthentiPad()	

2.2.1.10 Enter Registration Flow (P6)

The Registration Flow allows you to register a new image and caption, questions, and so on as described in the table below:

Table 2–12 Registration Flow

Module	APIs	Description
Server	VCryptRulesEngine::processRules()	For method details, see Section 4.6.1, "processRules."
Oracle Adaptive Access Manager Sample	registerImagePhrase.jsp	Assigns new image and caption to user
		Assigns new image and caption to user
		Forwards to page handleRegisterImagePhrase.jsp
	registerQuestions.jsp	Gets question pick set for the user
		Displays question selection user interface and inputs for answers
		Forwards to page handleRegisterQuestions.jsp
	registerContactInfo.jsp	Presents user with inputs for OTP registration information
		Forwards to page handleRegisterContactInfo.jsp
BharosaHelper	BharosaHelper::getAuthentiPad()	
	BharosaHelper::createSampleAuthentiPad	
	BharosaHelper::assignRandomImageAndCaption	
	BharosaHelper::saveNewImageAndOrCaption	
	BharosaHelper::getQuestions	
	BharosaHelper::isDeviceRegistered	
	BharosaHelper::setContactInfo	

2.2.1.11 Run Challenge Rules (R5)

The challenge rules are invoked to determine which type of challenge to display to the user. Values returned by the challenge rules include the following:

- **ChallengeQuestion** to challenge the user with question.
- **ChallengeSMS** to challenge user with OTP via SMS, to challenge user with OTP
- **ChallengeEmail** to challenge user with OTP via email
- **Block** to block the user.

Table 2–13 lists the APIs used to run the challenge rules.

Table 2–13 Run Challenge Rules APIs

Module	APIs	Description
Server	VCryptRulesEngine::processRules()	For method details, see Section 4.6.1, "processRules."
Oracle Adaptive Access Manager Sample	handleChallenge.jsp	handleChallenge.jsp calls BharosaHelper::validateAnswer If that method returns BharosaEnumChallengeResult.SUCCESS, status is updated to "success" and the user is allowed to move forward; otherwise if BharosaEnumChallengeResult.WRONG_ANSWER is returned then challenge rules are run again to determine the next step.
BharosaHelper	BharosaHelper::validateAnswer()	

2.2.1.12 Run Authentication Rules (R6)

`BharosaHelper::getAuthentiPad` is used to create an authentication device. That method in turn calls the Authentication Device Rules to determine the device to use.

If the user is to be challenged with a question, the rule returns the QuestionPad. If the user is to be challenge with an OTP, the rule returns the TextPad.

2.2.1.13 Challenge the User (S6)

If appropriate, the user is challenged with either Knowledge Based Authentication (KBA) or OTP (One Time Password).

KBA is an extension to existing User ID/password authentication and secures an application using a challenge/response process where users are challenged with questions. The user must answer the question correctly to proceed with his requested sign-on, transaction, service, and so on.

OTP is an extension to existing User ID/password authentication as well and adds an extra security layer to protect applications. OTP is generated after verifying the user ID and password and then delivered to users via e-mail or mobile phone if the application deems it to be necessary. Users then use the OTP to sign-in to the application.

Table 2–14 lists the APIs to challenge the user with registered questions.

Table 2–14 Challenge User APIs

Module	APIs	Description
Server	VCryptAuth::getSecretQuestion() VCryptTracker::generateOTP()	
Oracle Adaptive Access Manager Sample	Challenge.jsp	Determine type of challenge to use. BharosaHelper::runChallengeRules If challenge type returned is KBA (ChallengeQuestion) then get user question with VCryptAuth:getUserQuestion If challenge type is OTP (ChallengeSMS, ChallengeEmail, ...) then generate, store, and send OTP code. ■ BharosaHelper::generateOTP ■ BharosaHelper::sendCode Use authentication pad rules to determine authentipad to display to the user. See Section 2.2.1.4, "Run Virtual Authentication Device Rules (R2).". Submits the answer to handleChallenge.jps handleChallenge.jsp collects user input and calls BharosaHelper::validateAnswer - used to validate user answer for challenge (same as question challenge)
BharosaHelper	BharosaHelper:: createPersonalizedAuthentiPad () BharosaHelper::createAuthentiPad() BharosaHelper::generateOTP BharosaHelper::sendCode BharosaHelper::getUserQuestion	
Client	AuthentiPad::getHTML()	

2.2.1.14 Check Answers to Challenge (C3)

This stage involves validating the user's input to the challenge:

- For KBA, calling Oracle Adaptive Access Manager Server to determine whether the answer the user has supplied matches the registered reply.
- For OTP, validating the entered value to the OTP generated and sent to the user.

Table 2–15 lists the APIs used to validate a challenge.

Table 2–15 Validate Answer to a Challenge

Module	APIs	Description
Server	VCryptAuth::authenticateQuestion() VCryptRulesEngine::processRules() VCryptTracker::updateAuthStatus()	For method details, see Section 4.6.1, "processRules," and Section 4.5.9, "updateAuthStatus."
Oracle Adaptive Access Manager Sample	handleChallenge.jsp	Calls BharosaHelper::validateAnswer If that method returns BharosaEnumChallengeResult.SUCCESS, status is updated to "success" and the user is allowed to move forward; otherwise if BharosaEnumChallengeResult.WRONG_ ANSWER is returned then challenge rules are run again to determine the next step.
BharosaHelper	BharosaHelper:: validateAnswer()	If the type of challenge being validated is KBA (ChallengeQuestion), then VCryptAuth::authenticateQuestion is called to validate the users input against the registered answer for the question presented. If the type of challenge being validated is OTP (ChallengeSMS, ChallengeEmail, and so on), then the users input is compared to the value stored when OTP code was generated. If the answer is correct, the OTP challenge counter is reset by calling BharosaHelper::resetOTPCounter. Otherwise if the answer is incorrect, the OTP challenge counter is incremented (BharosaHelper::incrementOTPCounter). Method returns a BharosaEnumAuthStatus of either BharosaEnumAuthStatus.SUCCESS or BharosaEnumAuthStatus.WRONG_ANSWER

2.2.1.15 Lock Out Page (S2)

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

2.2.1.16 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 2–7 illustrates a flow of authentication that uses this solution. For details about the stages of this flow, see the following sections:

2.2.2.1 User/Password (S1)

The User/Password Page is the existing page currently used by the client. It contains the text box for both the username and password. There are no changes required for

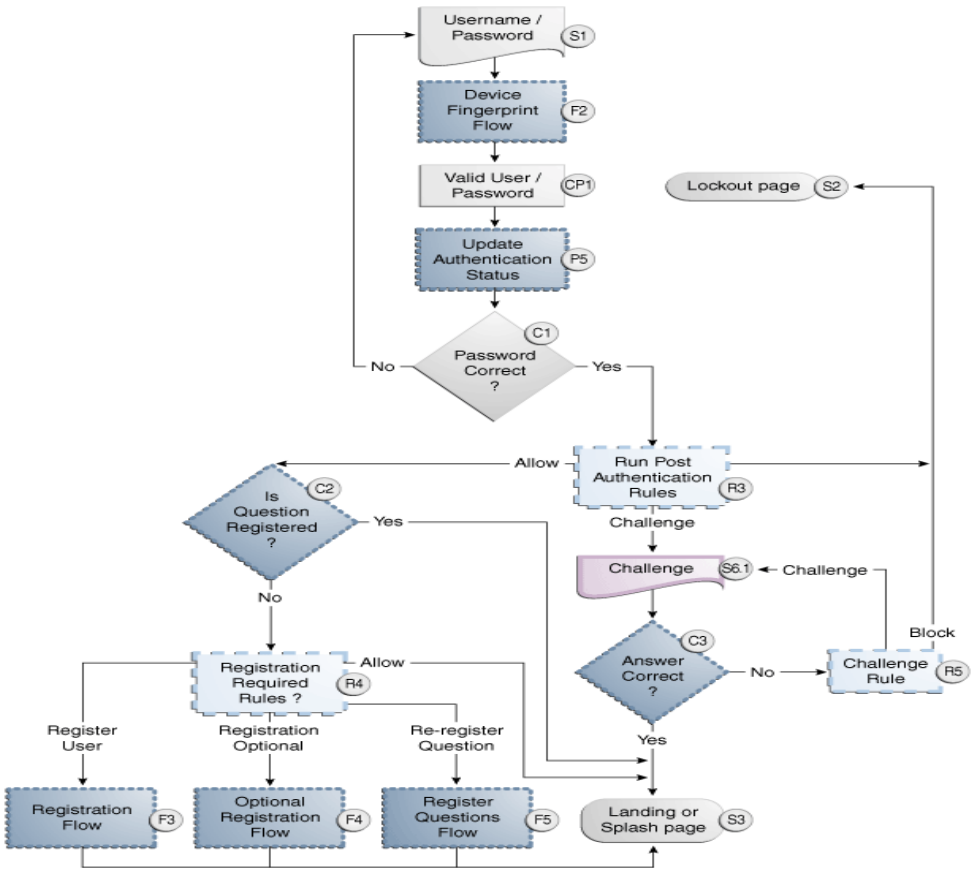
this page; however, the post from this page should display a transient (intermediate) refresh page.

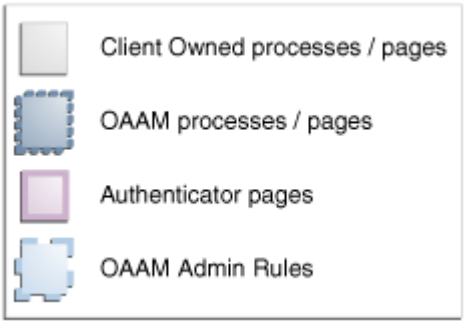
2.2.2.2 Stages

For information on the other stages, see the following sections:

- Section 2.2.1.2, "Device Fingerprint Flow (F1)"
- Section 2.2.1.6, "Validate User and Password (CP1)"
- Section 2.2.1.6.1, "Update Authentication Status (P5)"
- Section 2.2.1.6.2, "Password Status (C1)"
- Section 2.2.1.7, "Run Post-Authentication Rules (R3)"
- Section 2.2.1.8, "Check Registration for User (C2)"
- Section 2.2.1.9, "Run Registration Required Rules (R4)"
- Section 2.2.1.13, "Challenge the User (S6)"
- Section 2.2.1.15, "Lock Out Page (S2)"
- Section 2.2.1.16, "Landing or Splash Page (S3)"

Figure 2–7 Knowledge-Based Authentication Scenario




Client Owned processes / pages
OAAM processes / pages
Authenticator pages
OAAM Admin Rules

3

Integrating Native .NET Applications

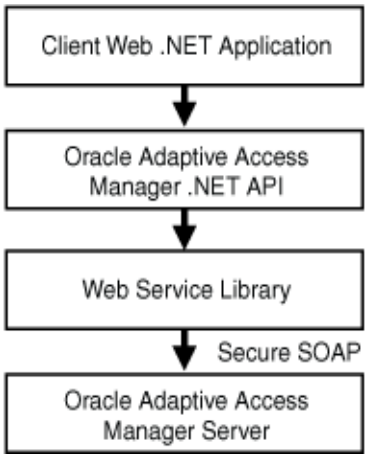
This chapter provides details how ASP.NET applications can integrate with Oracle Adaptive Access Manager using the .NET API provided by Oracle Adaptive Access Manager. Descriptions are also provided on the sample applications used to illustrate the integration of different OAAM features with a basic Web application.

This chapter contains the following sections:

3.1 Introduction

ASP.NET is a web application framework that allows programmers to build dynamic Web sites, web applications and web services. ASP.NET applications, written in any ASP.NET language, can use the OAAM .NET API to call Oracle Adaptive Access Manager. This API communicates with the OAAM server using Simple Object Access Protocol (SOAP), as illustrated in Figure 3–1.

Figure 3–1 .NET Application



3.2 Oracle Adaptive Access Manager .NET SDK

The Oracle Adaptive Access Manager .NET development kit (SDK) is packaged in the ZIP file, `oaam_native_dot_net.zip` in `$ORACLE_HOME/oaam/oaam_libs/dotNet/`.

Sample .NET applications that enable OAAM features require the integration of the OAAM .NET APIs found in the SDK package `oaam_native_dot_net.zip`. The content of the archive needs to be extracted to the root directory of the web application:

`oaam_native_dot_net.zip could be obtained from ${ORACLE_HOME}/oaam/dist/oaam_dist_final/oracle.oaam.libs/dotNet.`

3.3 Configuration Properties

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

3.3.1 How the API Uses Properties

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. Virtual authentication devices are controls for user input and provide virtual keyboard and personalization. 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 .NET Property Files

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

Table 3–1 (Cont.) .NET Property Files

Directory	Example
<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/

3.3.2 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==
```

3.3.3 Using User-Defined Enumerations to Define Elements

Visual Studio 2005 allows you to use enumerations defined in the .NET Framework. A user-defined enumerations are 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");
```

3.4 Oracle Adaptive Access Manager API Usage

This section contains details on how OAAM APIs are used to support common OAAM scenarios. You can also refer to the sample applications for details.

3.4.1 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 succssfully 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 3.5.1, "ASP.NET Applications."

3.4.2 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);
}
```

3.4.3 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 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
}
```

3.4.3.1 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
  ```

3.4.3.2 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);
```

3.4.4 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);

// Retrive 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 3.5.1, "ASP.NET Applications."

3.4.5 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);
```

3.4.6 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

3.4.6.1 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;
}
```

3.4.6.2 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);
   ```

3.4.6.3 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
}
```

3.4.7 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)

3.4.8 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>
```

3.5 Integration Example Using Sample Applications

This section shows you how to integrate an application through using one of the sample applications provided in the SDK.

3.5.1 ASP.NET Applications

The following four ASP.NET applications are included in this sample package to demonstrate integration of various OAAM 11g features in ASP.NET based applications.

Table 3–2 ASP.NET Applications

Application Name	Description
SampleWebApp	This is a basic ASP.NET application without OAAM integration. This application is provided so that the reader can easily see incremental changes required to integrate various OAAM feature, such as, tracker, authenticator, and KBA.
SampleWebAppTracker	This application demonstrates integration of OAAM tracker functionality to SampleWebApp listed above.
SampleWebAppAuthTracker	This application demonstrates integration of OAAM tracker and authenticator functionalities to SampleWebApp listed above.
SampleKBATracker	This application demonstrates integration of OAAM tracker and KBA functionalities to SampleWebApp listed above.

3.5.2 Sample Application Details

Details about the four applications are provided in this section.

3.5.2.1 SampleWebApp

This application contains the following pages that demonstrate a web application before OAAM integration.

1. LoginPage.aspx
   - Collects the user name and password using a simple HTML form.
   - Validates the login and password information
   - Depending upon the validation result, the user will be redirected to either Success.aspx or to LoginPage.aspx with appropriate error message
2. Success.aspx
   - Displays 'Successfully logged in' message with a link for logout
3. LogoutPage.aspx
   - Logs out the user session and redirects to login page

3.5.2.2 SampleWebAppTracker

This application contains the following pages that demonstrate integration of OAAM tracker functionality to the sample application listed above.

This application requires the integration of the OAAM .NET APIs found in the SDK package oaam_native_dot_net.zip. The content of the archive needs to be extracted to the root directory of the web application.

1. LoginPage.aspx
   - Collects the username and password using simple HTML form
   - Saves the login and password in the session
   - Redirects the user to LoginJumpPage.aspx to collect the flash finger print of the user device
2. LoginJumpPage.aspx
   - Loads the user from ARM (Adaptive Risk Manager) by calling AppUtil.InitUser() (AppUtil is included in the SDK package). If the user is not found, a new user record will be created

- Returns HTML to load flash object bharosa_web/flash/bharosa.swf in the browser. The flash object calls CookieManager.aspx (included in the SDK package) with flash finger print details. CookieManager.aspx records the finger print in ARM and in return sets a flash cookie on the user's device
- After a brief wait (to allow time to get the flash cookie from ARM), redirects the browser to LoginHandlerPage.aspx

3. LoginHandlerPage.aspx
   - Records the user login attempt with ARM by calling AppUtil.InitTracker()
   - Validates the login and password information
   - Updates ARM with the password validation status (success/wrong user/wrong password/disabled user, etc) by calling AppUtil.UpdateAuthStatus()
   - If password validation succeeds, runs post-authentication rules by calling AppUtil.RunPostAuthRules()
   - If the post-authentication rules return block, blocks the user login after updating ARM with this information
   - Depending upon the validation result and/or the rules result, redirects the user to either Success.aspx or to LoginPage.aspx with appropriate error message
4. Success Page
   - Displays 'Successfully logged in' message with a link for logout
5. Logout Page
   - Logs out the user session and redirects to login page

3.5.2.3 SampleWebAppAuthTracker

This application contains the following pages that demonstrate integration of OAAM authenticator and tracker functionalities to the sample application listed above. This application collects the password using authenticators offered by OAAM.

This application requires the integration of the OAAM .NET APIs found in the SDK package oaam_native_dot_net.zip. The content of the archive needs to be extracted to the root directory of the web application.

1. LoginPage.aspx
   - Collects the username using simple HTML form
   - Saves the login in the session
   - Redirects the user to LoginJumpPage.aspx to collect the flash finger print of the user device
2. LoginJumpPage.aspx
   - Loads the user from ARM (Adaptive Risk Manager) by calling AppUtil.InitUser() (AppUtil is included in the SDK package). If the user is not found, a new user record will be created
   - Returns HTML to load flash object bharosa_web/flash/bharosa.swf in the browser. The flash object calls CookieManager.aspx (included in the SDK package) with flash finger print details. CookieManager.aspx records the finger print in ARM and in return sets a flash cookie on the user's device

- After a brief wait (to allow time to get the flash cookie from ARM), redirects the browser to LoginHandlerPage.aspx

3. LoginHandlerPage.aspx
   - Records the user login attempt with ARM by calling AppUtil.InitTracker()
   - Redirects the user to PasswordPage.aspx to collect the password using OAAM authenticator.
4. PasswordPage.aspx

   On Load:

   a. Sets the session authentication status to 'Pending' in ARM
   b. Runs pre-authentication rules by calling the AppUtil.RunPreAuthRules()
   c. If the pre-authentication rules return block, blocks the user login after updating ARM with this information
   d. If the pre-authentication rules return allow, runs another set of rules to determine the authenticator to use for this user, by calling AppUtil.RunAuthentiPadRules()
   e. Creates appropriate authenticator by calling AppUtil.CreateAuthentiPad()and renders the authenticator into HTML by using the AppUtil.getAuthentiPadHTML(). The authenticator HTML would fetch the authenticator image by calling GetImage.aspx (included in the SDK package)
   f. Stores the authenticator object in the session for later use during image generation and password decode

   On PostBack:

   a. Decodes the password using the authenticator object stored in the session
   b. Validates the login and password information
   c. Updates ARM with the password validation status (success/wrong user/wrong password/disabled user, etc) by calling AppUtil.UpdateAuthStatus()
   d. If password validation succeeds, runs post-authentication rules by calling AppUtil.RunPostAuthRules()
   e. If the post-authentication rules return block, blocks the user login after updating ARM with this information
   f. Depending upon the validation result and/or the rules result, redirects the user to either Success.aspx or to LoginPage.aspx with appropriate error message
5. Success Page
   - Displays 'Successfully logged in' message with a link for logout
6. Logout Page
   - Logs out the user session and redirects to login page

3.5.2.4 SampleKBATracker

This application contains the following pages that demonstrate integration of OAAM authenticator, tracker and KBA (Knowledge Based Authentication) functionalities to the sample application listed above. This application shows authentication mechanisms using password and KBA authenticators offered by OAAM.

This application requires the integration of the OAAM .NET APIs found in the SDK package oaam_native_dot_net.zip. The content of the archive needs to be extracted to the root directory of the web application.

1. LoginPage.aspx
   - Collects the username using simple HTML form
   - Saves the login in the session
   - Redirects the user to LoginJumpPage.aspx to collect the flash finger print of the user device
2. LoginJumpPage.aspx
   - Loads the user from ARM (Adaptive Risk Manager) by calling AppUtil.InitUser() (AppUtil is included in the SDK package). If the user is not found, a new user record will be created
   - Returns HTML to load flash object bharosa_web/flash/bharosa.swf in the browser. The flash object calls CookieManager.aspx (included in the SDK package) with flash finger print details. CookieManager.aspx records the finger print in ARM and in return sets a flash cookie on the user's device
   - After a brief wait (to allow time to get the flash cookie from ARM), redirects the browser to LoginHandlerPage.aspx
3. LoginHandlerPage.aspx
   - Records the user login attempt with ARM by calling AppUtil.InitTracker()
   - Redirects the user to PasswordPage.aspx to collect the password using OAAM authenticator
4. PasswordPage.aspx

   On Load:

   a. Sets the session authentication status to 'Pending' in ARM
   b. Runs pre-authentication rules by calling the AppUtil.RunPreAuthRules()
   c. If the pre-authentication rules return block, blocks the user login after updating ARM with this information
   d. If the pre-authentication rules return allow, runs another set of rules to determine the authenticator to use for this user, by calling AppUtil.RunAuthentiPadRules()
   e. Creates appropriate authenticator by calling AppUtil.CreateAuthentiPad()and renders the authenticator into HTML by using the AppUtil.getAuthentiPadHTML(). The authenticator HTML would fetch the authenticator image by calling GetImage.aspx (included in the SDK package)
   f. Stores the authenticator object in the session for later use during image generation and password decode

   On PostBack:

   a. Decodes the password using the authenticator object stored in the session
   b. Validates the login and password information
   c. Updates ARM with the password validation status (success/wrong user/wrong password/disabled user, etc) by calling AppUtil.UpdateAuthStatus()

d. If the password validation fails, the user will be redirected to LoginPage.aspx with appropriate error message

e. If password validation succeeds, runs post-authentication rules by calling AppUtil.RunPostAuthRules()

f. The user will be taken through different flows, as shown below, depending upon the action from post-authenticator rules result:

Post-Authentication Action	Target URL
Block	LoginPage.aspx
Allow	Success.aspx
ChallengeUser	ChallengeUser.aspx
RegisterQuestions	RegisterQuestionsPage.aspx
RegisterUser	PersonalizationPage.aspx
RegisterUserOptional	PersonalizationPage.aspx

5. PersonalizationPage.aspx

- Introduces the user to device personalization explaining the steps that would follow to create a new Security Profile for the user
- If the post authentication rule returns RegistrationOptional, the user is allowed to skip the registration process by clicking the 'Skip' button to proceed to the Success.aspx page directly
- If registration is not optional, the user must register by clicking 'Continue' to proceed to the RegisterImagePhrase.aspx page

6. RegisterImagePhrase.aspx

- Allows the user to customize the randomly generated background image, caption and the type of security device used during authentication
- A new background image and caption is assigned by calling AppUtil.AssignNewImageAndCaption()
- The user selected security device is assigned by calling AppUtil.SetAuthMode()

7. RegisterQuestionsPage.aspx

- Displays sets of questions which the user can choose and register the correct answer for each.
- The sets of questions are fetched by calling proxy.getSignOnQuestions()

8. ChallengeUser.aspx

- Challenges the user by displaying a question-pad with one of the questions already registered by the user
- The answer is validated by calling proxy.authenticateQuestion() and the result is updated in ARM by calling AppUtil.UpdateAuthStatus()
- If the answer is wrong, a call to AppUtil.RunChallengeUserRules() is made and based on the result of which, the user will either be allowed to re-enter the answer or be redirected to the block page after updating the block status in ARM

- The number of attempts that a user gets to answer a question correctly is set by the rule administrator for ARM
- On successfully answering the question correctly, the user is forwarded to the Success.aspx page

9. Success Page
   - Displays 'Successfully logged in' message with a link for logout
10. Logout Page
    - Logs out the user session and redirects to login page

3.5.3 Setting Up the Environment

Source code for each application is placed in a directory of its own. Visual Studio Solution files for each of these applications can be found in the root directory. The four applications could either be run using Visual Studio 2005 or be deployed on Microsoft IIS 6.0 on Windows Server 2003. Solutions file 'SampleWebApps' can be used to load and view all applications together using Visual Studio.

Instuctions to set up the environment to successfully run the sample applications are provided in this section. After all the following have been applied, you should be able to run these sample applications and see how they integrates with OAAM 11g in different scenarios.

3.5.3.1 Modifying the web.config File

Ensure that Soap URL to access OAAM server is set correctly in `web.config` file of the application, as per your deployment configuration. An example is shown as follows:

```
<appSettings>
  <add key="BharosaSOAPURL"
       value="http://localhost:14300/oaam_server/services"/>
</appSettings>
<appSettings>
```

3.5.3.2 Setting Properties for Images

For sample applications integrating with OAAM 11g, set `bharosa.image.dirlist` in `bharosa_app.properties` to the path where "`oaam_images`" folder could be found. The "`oaam_images`" foloder is located at: `${ORACLE_HOME}/oaam/dist/oaam_dist_final/oracle.oaam.oaam_images`.

The folder name could be changed but then the path should be modified accordingly. For example, if all the files obtained from the path above is stored in a folder named `oaam_images` and this folder is put under the root directory of the web application. The path should be: `${Application_HOME}/oaam_images/`

Make sure `lookup.properties` is contained in `/bharosa_properties/` folder, which lists all the properties files that need to be read. It could be obtained from:

`${ORACLE_HOME}/oaam/apps/oaam_native/overrides/conf/bharosa_properties`

Find and comment out the `bharosa.authentipad.image.url` property.

3.5.3.3 Running the Application

For developers who have access to Microsoft Visual Studio 2005 to test the web applications, simply build the solution after making all the above changes and click "Debug->Start Debugging" in Visual Studio 2005.

For deployment of these applications, here are some tips to follow:

- System: Windows Server 2003
- Application server should be installed using ->Control Panel->Add or Remove Programs->Add/Remove Windows Components. IIS and ASP.NET should be enabled;
- Create "new website" using IIS manager by running "inetmgr" in command window;
- Make sure ASP.NET version is set to v2.0 through ASP.NET tab in website's "Properties";
- Make sure that ASP.NET v2.0 is set to "allowed" in IIS manager. If there is no ASP.NET v2.0 extension, add a new web service extension manually. Go to C:\WINDOWS\Microsoft.NET\Framework, there should be some folder named v2.0.50727 or similar if ASP.NET v2.0 is installed. Add v2.0.50727/aspnet_isapi.dll as a new web service extension;
- In "IIS Manager->Local Computer->Application Pools", open "Properties->Identity", simply select "Local System" on the right of "Predefined" option if you come across probelm accessing "C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\Temporary ASP.NET Files" when opening web application pages.

3.5.4 Example: Enable Transaction Logging and Rule Processing

The following pages demonstrate how to enable transaction logging and rule processing in OARM using the ASP.NET sample applications.

Prerequisites:

- Transaction definitions in Sample_Transaction_Defs.zip need to be available in OARM. Use 'Admin > Transactions > Import Transactions' to import the transaction definitions.
- Transaction models defined in models.zip should be available in OARM
- Following properties must exist in bharosa_app.properties at the OARM and the .NET client side:

```
tracker.transaction.status.enum=Enum for transaction status
tracker.transaction.status.enum.success=0
tracker.transaction.status.enum.success.name=Success
tracker.transaction.status.enum.success.description=Success
tracker.transaction.status.enum.block=1
tracker.transaction.status.enum.block.name=Block
tracker.transaction.status.enum.block.description=Block
tracker.transaction.status.enum.reject=2
tracker.transaction.status.enum.reject.name=Reject
tracker.transaction.status.enum.reject.description=Reject
tracker.transaction.status.enum.pending=3
tracker.transaction.status.enum.pending.name=Pending
tracker.transaction.status.enum.pending.description=Pending

profile.type.enum.pretransaction=70
```

```
profile.type.enum.pretransaction.name=PreTransaction
profile.type.enum.pretransaction.description=Pre Transaction
profile.type.enum.pretransaction.ruleTypes=user,device,location,in_session
profile.type.enum.pretransaction.listTypes=vtusers
profile.type.enum.pretransaction.finalactionrule=process_results.rule
profile.type.enum.pretransaction.isPreAuth=false

profile.type.enum.posttransaction=80
profile.type.enum.posttransaction.name=PostTransaction
profile.type.enum.posttransaction.description=Post Transaction
profile.type.enum.posttransaction.ruleTypes=user,device,location,in_session
profile.type.enum.posttransaction.listTypes=vtusers
profile.type.enum.posttransaction.finalactionrule=process_results.rule
profile.type.enum.posttransaction.isPreAuth=false
```

Transaction Page

- Dynamically generates the transaction type selection menu based on transaction enums defined in property file 'bharosa_common.properties'
- On selecting transaction type, dynamically renders the transaction fields based on field definitions defined in properties files.
- Either creates a transaction by calling AppUtil.createTransaction() or updates the transaction by calling AppUtil.updateTransaction()depending on the current form being submitted.
- Runs pre and post transaction rules by calling AppUtil.RunPreTransactionRules() or AppUtil.RunPostTransactionRules(). Depending upon the result, the browser is redirected to the next appropriate page.

4

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:

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 has to be deployed and targeted into the WebLogic Managed Server where the integrated application is deployed. Make sure the WebLogic Managed Server is part of the same WebLogic domain where OAAM is deployed.

4.1.1 Using Oracle Adaptive Access Manager Shared Library in Web Applications

Deploy the OAAM Web Applications Shared library `<IAM_HOME>/oaam/oaam_libs/war/oaam_native_lib.war` as a library.

To use the Oracle Adaptive Access Manager Shared Library in Web applications, 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 Using Oracle Adaptive Access Manager Shared Library in Enterprise Applications

Deploy the OAAM Enterprise Applications Shared library `<IAM_HOME>/oaam/oaam_libs/ear/oaam_native_lib.ear` as a library.

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

```
<library-ref>
```

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

4.1.3 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` folder of the native web application.

For instructions on customizing, extending, or overriding Oracle Adaptive Access Manager properties, refer to Chapter 7, "Customizing Oracle Adaptive Access Manager."

4.2 OAAM Java InProc Integration

Follow these steps:

1. Make sure you have set the reference to OAAM shared library "`oracle.oaam.libs`".
2. 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 the `WEB-INF\classes` folder of the native web application.
3. Set up OAAM Data Source with the JNDI name as "`jdbc/OAAM_SERVER_DB_DS`" and point it to the OAAM database.

4.3 OAAM SOAP Integration

To call the OAAM APIs via SOAP instead of inproc, follow these steps in these sections.

4.3.1 Set up SOAP Security

Setup SOAP User on WebLogic Server and OWSM Policy

Out-of-the-box, OAAM publishes Web services at the URL: `/oaam_server/services`. This URL is protected with HTTP Basic authentication.

Create a user that will be used for SOAP authentication, and add that user in the proper group. This user can access this URL. The user must be in the OAAMSOAPServicesGroup group.

To set up the OWSM Policy to set HTTP Basic Authentication on `/oaam_server/services` follow these steps:

1. Log in to Enterprise Manager using the URL `http://weblogic-admin-hostname:port/em`.
2. Under `weblogic_domain`, select the domain and select `oaam_server_server1` under that and right click and select the '`Web Services`' option.
3. Click the "`Attach Policies`" link in top right area.
4. Select all the rows corresponding to OAAM Web Services and click the **Next** button
5. To enable SOAP Authentication:

a. Select the row "**oracle/wss_http_token_service_policy**".

6. To disable SOAP Authentication:

   a. Select the rows "**oracle/binding_authorization_permitall_policy**", "**oracle/no_authentication_service_policy**", "**oracle/no_authorization_service_policy**" and click the **Next** button

7. Click the **Attach** button in the next page.

8. Restart OAAM Server if required.

Client Side Keystore to secure the SOAP User password

Web Services/SOAP clients need to send the username and password for successful communication with OAAM web services.

1. In the *$ORACLE_HOME*`/oaam/cli` directory, create a file, for example, `soap_key.file`, and enter the HTTP authentication user password in it. (The password from the user that was added to the OAAMSOAPServicesGroup role/group).

2. Copy `sample.soap_3des_input.properties` to `soap_3des_input.properties`.

3. Update `soap_3des_input.properties` with the keystore password, the alias password, and password file.

```
#This is the password for opening the keystore.
keystorepasswd=

#This is the password reading alias (key) in the keystore
keystorealiaspasswd=

#File containing from key. Please note, keys in AES could be binary. Also note
algorithms like 3DES require minimum 24 characters in the key
#keyFile=soap_key.file
keyFile=
```

4. Generate the keystore.

   - For Unix/Linux, run

```
$JAVA_EXE -Djava.security.policy=conf/jmx.policy -classpath $CLSPTH
com.bharosa.vcrypt.common.util.KeyStoreUtil updateOrCreateKeyStore
readFromFile=soap_3des_input.properties
```

   - For Windows, run

```
genkeystore.cmd soap_3des_input.properties
```

   If the `KeyStore` command was successful, you will see output similar to the following:

```
updateOrCreateKeyStore done!
Keystore file:system_soap.keystore,algorithm=DESede
KeyStore Password=ZG92ZTEyMzQ=
Alias Password=ZG92ZTEyMw==
```

5. Note down the Keystore password and Alias Password print on the screen. You will need to add these to `bharosa_server.properties`.

6. Save the `system_soap.keystore` file in your source code control system. Please take adequate security precaution while handling this file. The file contains critical password information. Make sure that only authorized personnel have read access

to this file. If you lose it, Oracle Adaptive Access Manager will not be able to recover data encrypted.

7. Copy your `system_soap.keystore` to the following directories:
   - `<application>/WEB-INF/classes/bharosa_properties` (classpath of the native client deployment)
   - `<OAAM application>/bharosa_properties`
8. Delete both the `soap_key.file` and `soap_3des_input.properties` files.
9. Add the following properties with the encoded passwords (from step 5) and the authentication username to `bharosa_server.properties`.

```
vcrypt.soap.auth.keystorePassword=<base64 encoded keystore password>
vcrypt.soap.auth.aliasPassword=<based64 encoded password to the alias>
vcrypt.soap.auth.username=<user configured for accessing the soap services>
vcrypt.soap.auth.keystoreFile=system_soap.keystore
```

4.3.2 Set SOAP Related Properties in bharosa_server.properties

Set the following properties in `bharosa_server.properties` of the native application:

```
vcrypt.common.util.vcryptsoap.impl.classname=com.bharosa.vcrypt.common.impl.VCrypt
SOAPGenericImpl
vcrypt.tracker.impl.classname=com.bharosa.vcrypt.tracker.impl.VCryptTrackerSOAPImp
l
vcrypt.user.image.dirlist.property.name=bharosa.image.dirlist
bharosa.config.impl.classname=com.bharosa.common.util.BharosaConfigPropsImpl
bharosa.config.load.impl.classname=com.bharosa.common.util.BharosaConfigLoadPropsI
mpl
vcrypt.tracker.soap.useSOAPServer=true
vcrypt.soap.disable=false
vcrypt.soap.auth.keystoreFile=system_soap.keystore


# Environment specific values need to be replaced below this line
vcrypt.tracker.soap.url=http://<host-name>:<port>/oaam_server/services
bharosa.image.dirlist=<absolute folder path where OAAM images are available>

# If SOAP Authentication is enabled, then the following have to be set
# otherwise just set the property vcrypt.soap.auth=false
vcrypt.soap.auth=true
vcrypt.soap.auth.keystorePassword=<Java keystore password>
vcrypt.soap.auth.aliasPassword=<Keystore alias password>
vcrypt.soap.auth.username=<SOAP User name>
```

4.4 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.

4.5 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

4.5.1 handleTrackerRequest

handleTrackerRequest captures fingerprint details and identifies 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 the machine 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. The corresponding enum name is `auth.client.type.enum`.
secureClientVersion	The version of the client; optional
digitalCookie	The digital signature cookie; it can be the flash cookie; it is 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; it can be the version of the flash client
fingerPrintType	Refer to the OAAM enum `vcrypt.fingerprint.type.enum` for a list of valid values. Currently the enum has following values: ■ browser=1 ■ flash=2 It is recommended to use `1` (for browser) as the value of `fingerPrintType` as this parameter corresponds to the browser fingerprint type
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; it is defined in the properties file; optional
fingerPrint2	The second fingerprint value; optional
requestTime	The time at which the request was made

4.5.2 createTransaction

createTransaction creates a new transaction.

```
public VCryptResponse createTransaction(
                TransactionCreateRequestData transactionCreateRequestData);
```

Table 4–2 createTransaction Parameter and Returned Value

Parameter	Description
TransactionCreateRequestData	The object to create a new transaction; it throws the exception `BharosaException` if it fails validation. The structure of this object is as follows: ■ **requestId** identifies the user session; required ■ **requestTime** is the time of the request; can be null; if null, the server uses the current time ■ **transactionKey** is the key to the transaction definition; used to create a transaction definition; required ■ **externalTransactionId** is used to correlate the application transaction with the corresponding OAAM Transaction. It can also be used to update the transaction. ■ **status** is the transaction status; can be null. The corresponding enum name is tracker.transaction.status.enum. ■ **contextMap** is the map of key-value pairs. Keys of this map should exactly match the Internal ID of the related Source Data of the Transaction Definition. The value should be always a java String value. If the value is a Date value then it should be in the format `yyyy-MM-dd'T'HH:mm:ss.SSSz"`
VCryptResponse	The response object; make sure to check `isSuccess()` before obtaining the transaction ID with the method `getTransactionResponse()`

4.5.3 updateTransaction

updateTransaction updates a previously created transaction.

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

Table 4–3 updateTransaction Parameter and Returned Value

Parameter	Description
TransactionUpdateRequestData	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 createTransaction. it throws the exception BharosaException if it fails validation. The structure of this object is as follows: ■ 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 is used to correlate the application transaction with the corresponding OAAM Transaction. It can also be used to update the transaction. ■ contextMap is a map of key-value pairs. Keys of this map should exactly match the "Internal ID" of the related "Source Data" of the Transaction Definition. The value should be always a java String value. If the value is a Date value then it should be in the format "yyyy-MM-dd'T'HH:mm:ss.SSSz".
VCryptResponse	The response object; make sure to check isSuccess() before obtaining the transaction ID with the method getTransactionResponse()

4.5.4 handleTransactionLog

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

4.5.5 updateTransactionStatus

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
Status	The transaction status
transactionId	The ID of the transaction with 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."

4.5.6 updateLog

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
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 `auth.status.enum`
clientType	An enumeration value indicating the client type used for authentication. The corresponding enum name is auth.client.type.enum.
clientVersion	The version of the client; optional
fingerPrintType	Refer to the OAAM enum `vcrypt.fingerprint.type.enum` for a list of valid values. Currently the enum has following values: ■ browser=`1` ■ flash=`2` It is recommended to use `1` (for browser) as the value of `fingerPrintType` as this parameter corresponds to browser fingerprint type.

Table 4–6 (Cont.) updateLog Parameters

Parameter	Description
fingerPrint	The fingerprint; if it describes browser characteristics, then the header is parsed into this string; it represents the browser header information
digFingerPrintType	Refer to the OAAM enum `vcrypt.fingerprint.type.enum` for list of valid values. Currently the enum has following values: ■ browser=1 ■ flash=2 It is recommended to use `2` (for flash) as the value of `digFingerPrintType`, as this parameter corresponds to flash fingerprint type.
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

4.5.7 getUserByLoginId

getUserByLoginId returns 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

4.5.8 generateOTP

VcryptTrackerImpl::generateOTP returns OTP code based on the following properties (to determine length of code returned and characters to use in creating OTP code)

`bharosa.uio.default.otp.generate.code.length`

`bharosa.uio.default.otp.generate.code.characters`

Example code for API use is in the OAAM example application available on Oracle by Example.

```
(String requestId, String challengeType, String appId)
```

Table 4–8 generateOTP

Parameter	Description
requestId	OAAM Request ID
challengeType	OAAM Challenge Type configured by the user defined enum: bharosa.uio.default.challenge.type.enum. For more information, refer to Section 11.7, "Registering SMS Processor to Perform Work for Challenge Type."
appId	An application identifier used to look up properties based on application. If no application specific properties are required, an empty string, null, or "default" can be passed.

4.5.9 updateAuthStatus

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 log in 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 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 4–9 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 `auth.status.enum`
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."

4.5.10 processPatternAnalysis

processPatternAnalysis triggers the data pattern processing.

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

Table 4–10 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 parameter is required.
status	A value of the user-defined enumeration `auth.status.enum`. 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.

4.5.11 markDeviceSafe

markDeviceSafe marks the user device as safe.

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

Table 4–11 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

4.5.12 IsDeviceMarkedSafe

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

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

Table 4–12 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

4.5.13 clearSafeDeviceList

clearSafeDeviceList clears the user safe device list of the user associated with a request.

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

Table 4–13 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.

4.6 Rules Engine

The Rules Engine is the part of the OAAM that enforces policies at checkpoint. OAAM 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.

4.6.1 processRules

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);
```

processRules calls the methods related to the Rules Engine, gets an instance of the Rules Engine by calling the method **VCryptTrackerUtil.getVCryptRulesEngineInstance()**.

Table 4–14 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. The runtimeTypes is a singleton list of Integer type. Refer to the "Information about execution of multiple checkpoints in the processRules() method" section below. For example, to run a pre-authentication checkpoint, create the following list: `List PRE_AUTH_RUNTIME_LIST = Collections.singletonList(new Integer(1));`
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

Information about execution of multiple checkpoints in the processRules() method

1. The order of checkpoint evaluation is based on the order of those in the List. The OAAM Rules Engine iterates over the list of checkpoints and evaluates one checkpoint at a time.
2. The result of each checkpoint evaluation is stored into ResultMap with CheckPointId as the key and `VCryptRulesResult` as the value.
3. The `ResultMap` is then set onto `VCryptRulesResult`.
4. `VCryptRulesResult` is returned as the result of `processRules()` method.
5. If there is a failure in execution of any checkpoint, the corresponding `VCryptRulesResult` in `ResultMap` will capture that information, but the execution of other checkpoints is not impacted. However, if there is a system failure, then the result of `processRules()` itself will have the details of the error.

It is recommended to test the success status of result from `processRules()` method before the caller tries to fetch result of each checkpoint execution.

Getting Device ID

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 = rulesResult.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.devideId` is set to true, so the device ID can be captured:

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

Valid Checkpoints

For list of valid checkpoints, refer to the OAAM enumeration `profile.type.enum`. For example `profile.type.enum.preauth=1` indicates that the Pre-Authentication checkpoint is indicated using the numeric value `1`.

Location and Device Data

With property `bharosa.tracker.sendLocationData=true` set, location (city, state, country names) 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) ;
}
```

4.7 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

4.7.1 getFinalAuthStatus

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 4–15 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

4.7.2 setTemporaryAllow

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 4–16 setTemporaryAllow Parameters

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

4.7.3 cancelAllTemporaryAllows

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

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

Table 4–17 cancelAllTemporaryAllows Parameters

Parameter	Description
customerId	The customer ID

4.7.4 resetUser

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

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

Table 4–18 resetUser Parameters

Parameter	Description
customerId	The customer ID

4.7.5 getRulesData

getRulesData returns all rules executed for the given session ID and provides information about the rules that were triggered.

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

Table 4–19 getRulesData Parameters

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

4.7.6 getActionCount

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 4–20 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 `actionEnum`; required. The corresponding enum name is rule.action.enum.

Note: For this API to work, the corresponding action `incrementCacheCounter` enum property needs to be set to `true`.

5

Native API for OTP Challenge

Oracle Adaptive Access Manager's Native OTP API offers a way to add another factor to a traditional user name/password authentication scheme.

This chapter contains the following information:

5.1 OTP Integration Overview

Native OTP Challenge integration enables strong authentication for access to applications.

> **Note:** For information about administrative tasks you can perform for OTP such as resetting OTP profiles, unlocking users, viewing OTP case details, and viewing OTP performance data, see the *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*.

The flow of interaction is as follows: When the User ID and password are successfully verified, if the application deems it to be necessary, a one time password is sent to the user's mailbox or mobile phone. This one time password will be verified and only then will the user be authenticated to the application.

> **Note:** The application authenticates the OTP code given by the user through custom implementation.

5.1.1 One Time Password (OTP)

One Time Password (OTP) is a random single use authentication credential. The OTP may be either numeric or alphanumeric and any length and the randomization algorithm is pluggable.

The following are major benefits of using out-of-band OTP:

- The one time password is delivered to the valid user through one of the configured channels. These can include SMS, IM, email or voice.
- The user does not require any proprietary hardware or client software of any kind.

5.1.2 OAAM OTP Challenge Functionality

OAAM OTP challenge functionality allows the end user to register profile information for use as a communication channel subsequently to challenge the user if appropriate. The user is sent an email or SMS with a generated one time use password and presented with a challenge page in which he can enter the generated code.

Oracle Adaptive Access Manager offers an OTP code generation API that can be used by native integration APIs.

5.1.3 Sample

A sample application, OAAM Sample, is available as a form of documentation to illustrate a native implementation of an Oracle Adaptive Access Manager integration.

It includes registration and email challenge related flows that provide integrators with an example of how to use the OTP APIs for generating OTP code, incrementing the OTP challenge counter, and clearing the OTP challenge counter.

OAAM Sample implements example flows using JSPs to both display pages generate code, and handle the user input of pages, backed by the BharosaHelper utility class to make calls into the OAAM APIs for tracking user details and challenge statistics.

Note: Oracle Adaptive Access Manager ships with "oaam_native_lib.war" which must be deployed to run OAAM Sample.

5.2 OTP Registration and Challenge Experience

OAAM OTP challenge allows the end user to register profile information such as an email address or a mobile phone number or both for use as communication channel to challenge them.

The user is sent an email or SMS with a generated one time use password and presented with a challenge page in which they can enter the generated code.

The registration and challenge flows are presented in Section 5.3, "New User Registration" and Section 5.4, "User OTP Challenge."

5.3 New User Registration

Registration is the enrollment process, the opening of a new account, or other event where information is obtained from the user.

During the Registration process, the user is asked to register for questions, image, phrase and OTP (email, phone, and so on) if the deployment supports OTP. Once successfully registered, OTP can be used as a secondary authentication to challenge the user.

The login process begins with entering standard user name and password credentials. During a session, if the user is OTP-challenged, a single-use password is delivered to the user through the configured delivery channel he selected. The user retrieves the one-time password, then enters it.

In a new registration flow which include OTP:

- User Name Entered on Login Page
- Password Page is Presented and User Enters Password
- User Enters Registration Flow

- User Continues into Application

5.3.1 User Name Entered on Login Page

The user is presented with a page in which he is asked to submit his user name. The user name (login ID) is accepted from the first page and stored in the HTTP session. The user name page is followed by a transient page for capturing the flash and secure cookies and for fingerprinting the device.

For information on the JSPs, BharosaHelper utility class, and OAAM APIs used in this flow, refer to the following sections:

- User Name Page (S1)
- Device Fingerprint Flow (F1)
- Run Pre-Authentication Rules (R1)

5.3.2 Password Page is Presented and User Enters Password

The password page is displayed. The user fills in the password and clicks the Enter button on the device. Oracle Adaptive Access Manager verifies the user's password.

For information on the JSPs, BharosaHelper utility class, and OAAM APIs used in this flow, refer to the following sections:

- Run Virtual Authentication Device Rules (R2)
- Decode Virtual Authentication Device Input (P4)
- Validate User and Password (CP1)
- Run Post-Authentication Rules (R3)
- Check Registration for User (C2)
- Run Registration Required Rules (R4)
- Enter Registration Flow (P6)

5.3.3 User Enters Registration Flow

The user will continue through the registration process.

5.3.3.1 User selects an authentication pad background image

The user selects an anti-phishing image and phrase.

5.3.3.2 User registers challenge questions

The user selects challenge questions and enters the answers to those questions.

5.3.3.3 User Opts In to OTP

He agrees to register his profile for OTP challenge

5.3.3.4 User registers profile information

The user enters his profile information in profile registration page.

The user's contact information, such as mobile phone number and email address, is registered.

5.3.3.5 User Agrees to Terms and Conditions

User agrees to the terms and conditions presented on the registration page.

5.3.4 User Continues into Application

The user continues on to the application.

5.4 User OTP Challenge

An OTP challenge is when the user is asked to provide the OTP as a form of authentication for risk situations based upon configured policies.

The user must enter the correct OTP in to the Web interface to proceed with the operation.

In the challenge flow which includes OTP:

- User Name Entered on Login Page
- Password Page is Presented and User Enters Password
- OAAM Rules Determine User Should Be Challenged via OTP
- User Continues Into the Application

5.4.1 User Name Entered on Login Page

The user is presented with a page in which he is asked to submit his user name. The user name (login ID) is accepted from the first page and stored in the HTTP session. The user name page is followed by a transient page for capturing the flash and secure cookies and for fingerprinting the device.

For information on the JSPs, BharosaHelper utility class, and OAAM APIs used in this flow, refer to the following sections:

- User Name Page (S1)
- Device Fingerprint Flow (F1)
- Run Pre-Authentication Rules (R1)

5.4.2 Password Page is Presented and User Enters Password

The password page is displayed. The user fills in the password and clicks the Enter button on the device. Oracle Adaptive Access Manager verifies the user's password.

For information on the JSPs, BharosaHelper utility class, and OAAM APIs used in this flow, refer to the following sections:

- Run Virtual Authentication Device Rules (R2)
- Decode Virtual Authentication Device Input (P4)
- Validate User and Password (CP1)

5.4.3 OAAM Rules Determine User Should Be Challenged via OTP

The custom policies returns "Challenge" as an action, and the Challenge checkpoint determines that OTP is the type of challenge to be used.

For information on the JSPs, BharosaHelper utility class, and OAAM APIs used in this flow, refer to the following sections:

- Run Post-Authentication Rules (R3)
- Run Challenge Rules (R5)
- Run Authentication Rules (R6)
- Challenge the User (S6)
- Check Answers to Challenge (C3)

5.4.3.1 Generate OTP Code and Code is Delivered to the User through Custom Implementation

The system generates the OTP code and through custom implementation the code is delivered to the user.

The generateOTP API is used to generate OTP code. For information on this API, refer to Section 4.5.8, "generateOTP."

5.4.3.2 User Presented with Challenge Page

The user is presented with the challenge page, as shown in the example below.

The OTP Challenge devices are determined by the Authentication Pad checkpoint. The default device is TextPad.

For information on the Authentication Pad checkpoint, refer to Section 2.2.1.12, "Run Authentication Rules (R6)."

5.4.3.3 User Enters the Generated Code Sent to Him by the Application and is Validated by Custom Implementation

The user enters the generated code sent to him by the application and is validated by custom implementation.

5.4.4 User Continues Into the Application

The user continues into the application.

Part II

Universal Installation Option

Part II contains the following chapter:

6

Oracle Adaptive Access Manager Proxy

Oracle Adaptive Access Manager 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 UIO Proxy is available for the Apache Web server and Microsoft Internet Security and Acceleration (ISA) Server. In this chapter the Oracle Adaptive Access Manager Proxy for Apache will be referred to as the UIO Apache Proxy; and the Oracle Adaptive Access Manager Proxy for Microsoft ISA will be referred to as the UIO ISA Proxy.

This chapter:

- Explains the use and configuration of the UIO Proxy.
- Provides instructions for both Microsoft Internet Security and Acceleration (ISA) Server and Apache Web server implementations.

The intended audience is for integrators who configure the UIO 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:

For information on configuring OAAM Server, the client-facing multifactor authentication Web application specific to the UIO Proxy deployment, refer to Chapter 8, "Customizing the OAAM Server."

6.1 Introduction

The Introduction section of this chapter contains the following topics:

6.1.1 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 UIO 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 UIO 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 UIO Proxy redirects the client browser to OAAM Server for tracking and authentication purposes as defined by the UIO Proxy XML configuration.

6.1.2 Architecture

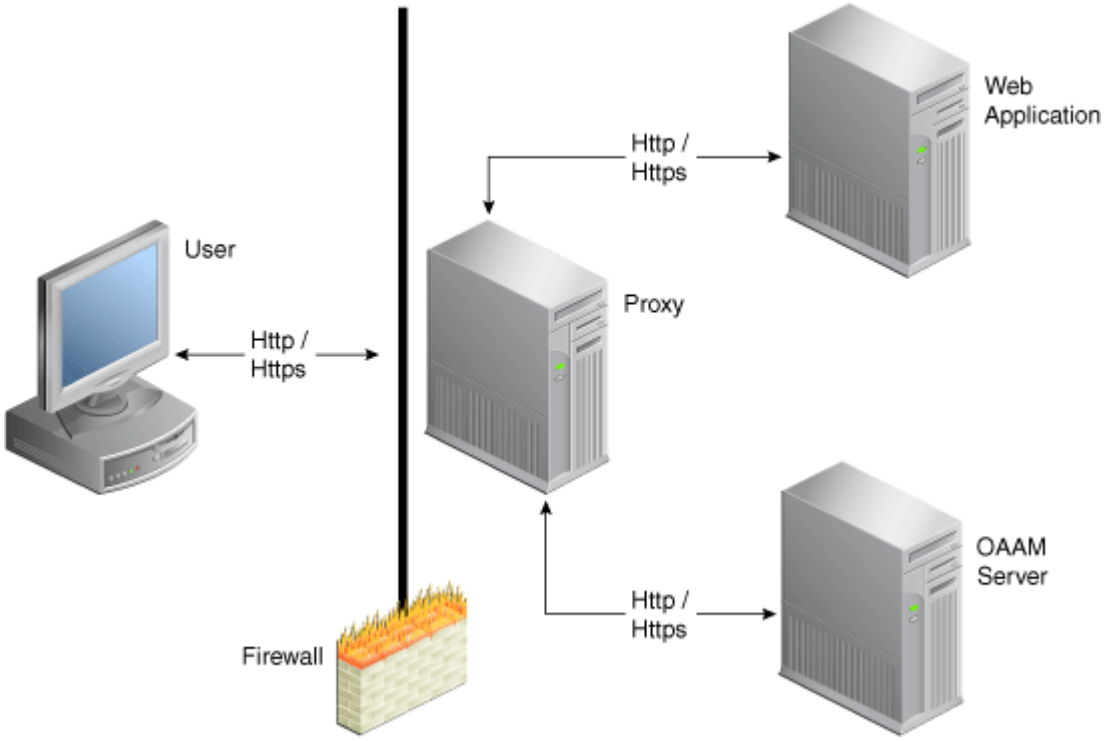
The following diagrams show a typical UIO Proxy deployment.

The first diagram shows a Web application before the UIO Proxy is deployed to provide multifactor authentication.

Figure 6–1 Before the Oracle Adaptive Access UIO Proxy

The second diagram shows various components added after the UIO Proxy deployment.

Figure 6–2 After UIO Proxy Deployment



The UIO 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.

6.1.3 References

For information on installing and configuring the Microsoft ISA server, refer to the refer to the relevant Microsoft documentation on Microsoft ISA Server setup. 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

6.2 Installing UIO ISA Proxy

The UIO ISA Proxy 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 UIO Proxy, a `.NET` framework 2.0 or better should to be installed. Install all the recommended updates from Microsoft on the machine.

Install Microsoft ISA Server 2006 Standard Edition and create Web publishing rules for the Web applications before installing the UIO 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 6.2.1, "UIO Proxy Web Publishing Configuration."
- Instructions on installation and programming information for the UIO ISA Proxy.
  - Section 6.2.2, "Registering the UIO ISA Proxy DLL."
  - Section 6.2.3, "Settings to Control the UIO Proxy."

6.2.1 UIO 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.

6.2.1.1 Web Listener Creation

For details on creating a Web listener, refer to the relevant Microsoft documentation. This section provides an outline.

1. For the Web Listener Name, enter **Bharosa Proxy Listener**.
2. Select **SSL** secure connection as the type of connection the Web listener establishes with clients.
3. For the Web Listener IP Addresses, choose **external**, **internal**, and **local host**.
4. Choose a single certificate for the Web Listener and select the certificate.
5. Specify that you do not want authentication for how clients validate their credentials.

6.2.1.2 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 UIO ISA Proxy, only the UIO Proxy machine, which runs Microsoft ISA Server, is accessible from the Internet.

Publish the following via Web publishing rules in the Microsoft ISA Server:

- OAAM Server

- Web applications

You need to set two (at least) rules: one for the main application and another for OAAM Server.

For detailed instructions on publishing rules, refer to the relevant Microsoft documentation. The following tips are provided for your reference.

6.2.1.2.1 Web Publishing Rule Creation for OAAM Server

To create a new Web publishing rule for OAAM Server you must 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 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.

6.2.1.2.2 Web Publishing Rule Creation for Protected Web Applications

To create a new Web publishing rule for Web applications, you must 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 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.

6.2.2 Registering the UIO ISA Proxy DLL

The UIO ISA Proxy is installed as a Web filter in Microsoft ISA Server. To install the UIO ISA Proxy, 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
```

6.2.3 Settings to Control the UIO Proxy

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

6.2.3.1 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 UIO Proxy configuration.
- The filename can either be fully qualified or relative to the location of the `BharosaProxy.dll`.
- The proxy loads 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 is loaded in the lexicographic order of the filenames in the registry.

- Changes under the `ConfigFiles` key are not effective until either the proxy is restarted or `HKLM\SOFTWARE\Bharosa\Proxy\ReloadConfig` is set to `1`.

6.2.3.2 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 are not applied until the `ReloadConfig` registry value is set to a nonzero value. When setting `ReloadConfig` to a nonzero value, the proxy loads configuration files. After loading the files, the proxy resets the `ReloadConfig` value to `0`.

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

6.2.3.3 Session ID Cookie

The UIO ISA Proxy 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 UIO ISA Proxy uses the cookie name, `BharosaProxy_SessionId`.

6.2.3.4 Configuring Session Id Cookie attributes via Global Variables

The attributes of the `Session Id Cookie` can be configured using global variables listed in Table 6–1.

Table 6–1 Session Id Cookie Attributes via Global Variables

Cookie Attribute	Global Variable Name	Description
expires	SessionCookie_ExpiryInMinutes	`expires` attribute is 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	SessionCookie_IsHttpOnly	`HttpOnly` attribute is added to the session cookie if this global variable is set to value greater than 0
secure	SessionCookie_IsSecure	`secure` attribute is added to the session cookie if this global variable is set to value greater than 0.
domain	SessionCookie_ DomainLevelCount	`domain` attribute is 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, the proxy uses 2 as the value.

6.2.3.5 Session Inactive Interval

Sessions in the UIO ISA Proxy are 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 UIO ISA Proxy removes a session after 1200 seconds (30 minutes) of inactivity. This value should be set to at least a few seconds higher than the Web application `session timeout` value.

6.2.3.6 Settings for Troubleshooting

Trace messages from the UIO ISA Proxy 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 6–2.

Table 6–2 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 renames 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 are written to file only if this value is nonzero.
TraceToDebugTerminal	REG_DWORD	Trace messages are 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 ignores all the interceptors defined in the UIO Proxy configuration. Essentially this places the UIO ISA Proxy 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 is necessary back up of all the HTTP traffic through the proxy. On such occasion, this registry value should be set to nonzero.

6.3 Installing UIO Apache Proxy

To install the UIO Apache Proxy, a new Apache Hypertext Transfer Protocol Daemon (**httpd**) has to be installed into which the UIO Apache Proxy is installed. This Apache httpd uses the **mod_proxy**, a module that implements the proxy/gateway/cache, to reverse-proxy (proxy on behalf of the backend application that has to be protected).

The Installation section contains information for installing the UIO Apache Proxy for Windows and Linux platforms.

The installation procedure involves:

- Ensuring that the Apache `httpd` requirements are met

  See Section 6.3.2, "Apache httpd Requirements."

- Copying the UIO Proxy `dlls` and supported `dlls` to specific directories in Apache

See Section 6.3.3, "Copying the UIO Apache Proxy and Supported Files to Apache."

- Configuring `memcache` (for Linux only)

  See Section 6.3.4, "Configuring Memcache (for Linux only)."

- Editing the httpd.conf to activate the UIO Proxy

  See Section 6.3.5, "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 UIO Proxy using application configuration XML files

  Section 6.3.6, "Modifying the UIO Apache Proxy Settings."

The post-installation procedures involve:

- Section 6.4, "Setting Up Rules and User Groups."

  Creating a new user to run the UIO Apache Proxy process (on Linux only)

- Section 6.5, "Setting Up Policies."

6.3.1 UIO Proxy Files for Windows and Linux

The UIO Apache Proxy binaries for Windows and Linux are different. Since the UIO Proxy is in C/C++, the same binary do not work on different platforms (unlike Java).

The files are located under *$ORACLE_HOME*`/oaam/oaam_proxy`*platform_specific_file*.

6.3.1.1 Windows

For Windows, the binary files are listed in Table 6–3.

Table 6–3 Windows Binary Files

Name	Description
mod_uio.so	UIO Apache Proxy module
log4cxx.dll	Apache Log4cxx library
libxml2.dll	XML/HTML Parser
apr_memcache.dll	APR Memcache client library.

The data files are listed in Table 6–4.

Table 6–4 Windows Data files

Name	Description
UIO_Settings.xml	UIO Apache Proxy Settings XML file
UIO_log4j.xml	UIO Apache Proxy Log4j (log4cxx) configuration XML file
TestConfig.xml	UIO Apache Proxy 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

6.3.1.2 Linux

For Linux, the binary files are listed in Table 6–5.

Table 6–5 Linux Binary Files

Name	Description
mod_uio.so	UIO Apache Proxy 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 6–6.

Table 6–6 Linux Data Files

Name	Description
UIO_Settings.xml	UIO Apache Proxy Settings XML file
UIO_log4j.xml	UIO Apache Proxy Sample Log4j (log4cxx) configuration XML file
TestConfig.xml	UIO Apache Proxy 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

6.3.2 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.

6.3.2.1 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

6.3.2.2 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

6.3.3 Copying the UIO Apache Proxy and Supported Files to Apache

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

6.3.3.1 Windows

Table 6–7 summarizes:

- The directories you have to copy the UIO Apache Proxy files to after installation
- The tree structure of the UIO Apache Proxy libraries and configuration files, assuming that you installed the files in `C:\Apache2.2`
- The directories the UIO Apache Proxy binary files go into are listed in Table 6–7.

Table 6–7 Directories for Windows UIO Proxy Binary Files

Directories	File Descriptions
C:\Apache2.2\modules\mod_uio.so	UIO Apache Proxy module
C:\Apache2.2\bin\log4cxx.dll	Apache Log4cxx library
C:\Apache2.2\bin\libxml2.dll	XML/HTML Parser
C:\Apache2.2\bin\apr_memcache.dll	APR Memcache library.

The data files are put in the directories summarized in Table 6–8.

Table 6–8 Directories for Windows UIO Proxy Data Files

Directories	File Descriptions
C:\OAAMUIO\UIO_Settings.xml	UIO Apache Proxy settings XML file
C:\OAAMUIO\UIO_log4j.xml	UIO Apache Proxy Log4j (log4cxx) configuration XML file
C:\OAAMUIO\TestConfig.xml	UIO Apache Proxy application configuration files (any number)
C:\OAAMUIO\UIO_Settings.rng	Relax NG grammar for UIO_Settings.xml
C:\OAAMUIO\UIO_Config.rng	Relax NG grammar for application configuration XML files
C:\OAAMUIO\logs\uio.log	UIO Apache Proxy log

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

6.3.3.2 Linux

After the installation of the Apache `httpd`, you must copy the UIO Apache Proxy binary files into (assuming Apache `httpd` is installed in `/usr/local/apache2`) the directories shown in Table 6–9.

Table 6–9 Directories for Linux UIO Proxy Binary Files

Directories	Description
/usr/local/apache2/modules/mod_uio.so	UIO Apache Proxy Module
/usr/local/apache2/lib/liblog4cxx.so.0.10.0.0	Apache Log4cxx Library
/usr/local/apache2/lib/libxml2.so.2.6.32	XML/HTML Parser

Table 6–9 (Cont.) Directories for Linux UIO Proxy Binary Files

Directories	Description
/usr/local/apache2/lib/libapr_memcache.so.0.0.1	APR Memcache client library.

Then, create soft links to the libraries as follows:

```
cd /usr/local/apache2/lib
ln -s liblog4cxx.so.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 `oaamuio` that is the checkpoint user for the UIO Apache Proxy. The proxy configuration and log files are accessible by this user. Ensure that only this user can access the log files. Assuming `/home/oaamuio` is the home directory for this user, the directory structure looks like the one presented in Table 6–10.

The UIO Apache Proxy data files should follow the directory structure shown in Table 6–10.

Table 6–10 Directories for Linux UIO Proxy Data Files

Directories	Description
/home/oaamuio/uio/UIO_Settings.xml	UIO Apache Proxy settings XML file
/home/oaamuio/uio/UIO_log4j.xml	UIO Apache Proxy Log4j (log4cxx) configuration XML file
/home/oaamuio/uio/TestConfig.xml	UIO Apache Proxy 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	UIO Apache Proxy 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.

6.3.4 Configuring Memcache (for Linux only)

This is an optional configuration that may be needed for Linux deployment of UIO Apache Proxy. The UIO Apache Proxy maintains a session for the user where it keeps local state such as session level variables for the user. On Windows, there is always a single process for Apache httpd server running and so this session information is local to the process. On Linux, you could have multiple Apache httpd server processes running which means the session information cannot be kept local to the process but needs to be centralized. In this case, memcached is used to hold the session

information. The following description is to identify when you must use memcached to hold the UIO Apache Proxy session information.

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: `httpd` with prefork MPM (single-threaded) or with worker MPM (multi-threaded). The MPM is built into the `httpd` and is not a run-time option.

With prefork MPM, `httpd` maintains a pool of single-threaded processes, where each request is handled by a single process. In this case, you must configure UIO Apache Proxy to use memcached.

With worker MPM, `httpd` maintains a pool of multithreaded processes, where every process could be handling multiple requests at a time. In this case, you can configure Apache httpd to launch a single process and avoid using memcached. However, the default configuration launches multiple processes and if you want to keep that unchanged, then you must configure UIO Apache Proxy to use memcached. Here is an example of httpd.conf example that can be used to configure a worker MPM to launch a single process.

```
# Following forces worker MPM to run 1 process (make sure mod_cgid is
# not loaded, otherwise it starts one more httpd process).
# Basically ThreadLimit=MinSpareThreads=MaxSpareThreads=MaxClients=ThreadsPerChild
# and StartServers=1. Setting MaxRequestsPerChild to 0 ensures that the process is
not
# bounced.

<IfModule mpm_worker_module>

ThreadLimit 150
StartServers 1
MinSpareThreads 150
MaxSpareThreads 150
MaxClients 150
ThreadsPerChild 150
MaxRequestsPerChild 0

</IfModule>
```

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 UIO Apache Proxy session data must be maintained in a common store (database or cache) so that multiple processes can access the session data. The UIO Proxy uses `memcache` (a memory based very fast cache) to store the session data.

At startup, the UIO 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 tries to connect to the `memcache` daemon using default connection parameters (that are defined in Section 6.3.6.1, "UIO_ Settings.xml"). On Windows, by default, the UIO Proxy uses local sessions. It does not connect to the `memcache` daemon; however it can also be configured to maintain session data in the `memcache` daemon (explained in Section 6.3.6.1, "UIO_ Settings.xml").

For the scenarios where the UIO Apache Proxy is connecting to `memcache` daemon, you must 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 UIO Apache Proxy has been tested with version 1.2.5 of memcache.

6.3.5 Configuring httpd.conf

This section provides information on how to edit the httpd.conf file to activate the UIO Apache Proxy. The httpd.conf file is the main configuration file used by the Apache HTTP Server.

6.3.5.1 Basic Configuration without SSL

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

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 UIO Apache Proxy.

```
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 UIO Apache Proxy.

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

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

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

On Linux,

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

4. Disable mod_proxy's forward-proxying 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 oaam_server and the target application is being protected by OAAM.

```
ProxyPass /oaam_server/
http://<FQDN_oaam_server>:<oaam_server_port>/oaam_server/
ProxyPassReverse /oaam_server/ http://<FQDN_oaam_server>:<oaam_server_
port>/oaam_server/

ProxyPass /<target_app>/ http://<FQDN_target_app>:<target_app_port>//<target_
appl>
ProxyPassReverse /<target_app>/ http://<FQDN_target_app>:<target_app_
```

```
port>/<target_app>
```

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 OAAM Server (`oaam_server`) and target application and run Phase One scenarios. The URL for the target application is:

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

So far in this chapter, the configuration to the proxy has been performed without using SSL.

6.3.5.2 Configuration with SSL

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

Note that the UIO Apache Proxy requires `mod_ssl` to be part of `httpd`. This ensures that the OpenSSL library is linked in and is properly configured for the UIO Apache Proxy to generate session ids. You need to ensure that `mod_ssl` is loaded and you do not need to perform 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 Web server `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 `<a href="http://appserver.example.com/foo/bar.html">foobar</a>` to `<a href="http://www.example.com/appserver/foo/bar.html">foobar</a>` making it accessible from outside."

6.3.6 Modifying the UIO Apache Proxy Settings

6.3.6.1 UIO_Settings.xml

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

```
 <!-- Log4jProperties location="/home/oaamuio/uio/UIO_log4j.xml"/ -->
       <Log4jProperties location="f:/oaamuio/uio/UIO_log4j.xml"/>
       <Memcache ipaddress="127.0.0.1" port="11211" maxconn="10"/>

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

       <ConfigFile location="/home/oaamuio/uio/TestConfig1.xml" enabled="true"/>
       <ConfigFile location="/home/oaamuio/uio/TestConfig2.xml" enabled="false"/>

       <ConfigFile location="f:/oaamuio/uio/TestConfig1.xml" enabled="false"/>

       <Setting name="GarbageCollectorInterval_ms" value="60000"/>
       <Setting name="MaxSessionInactiveInterval_sec" value="1200"/>
       <Setting name="CachedConfigExpiry_sec" value="120"/>
       <Setting name="SessionIdCookieName_str" value="SessionId"/>

       <Setting name="SessionCookie_ExpiryInMinutes" value="0"/>
       <Setting name="SessionCookie_IsHttpOnly" value="0"/>
       <Setting name="SessionCookie_IsSecure" value="1"/>

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


<!-- Enable AutoLoadConfig for Windows or Single-process Linux.
 Do not use for Multiple-process Linux when in production.
-->
<Setting name="AutoLoadConfig" value="1"/>

<!-- Setting name="UseMemcache" value="1"/ -->

       </UIO_ProxySettings>
```

Log4jProperties

Set the location of `log4j.xml` file that defines the logging configuration for the UIO Apache Proxy. 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 UIO 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 UIO Apache Proxy tries to connect to `memcache` on IP address `127.0.0.1` and port `11211`.

Settings

These are flags to control the behavior of the UIO Apache Proxy. Various settings are listed in Table 6–11.

Table 6–11 OAAM UIO Proxy Settings.

Flags	Description
MaxSessionInactiveInterval_sec	UIO Apache Proxy maintains a session for every user passes through the proxy. This setting sets the expiry time of this session after the user becomes inactive. It is in seconds (default is 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 updates the configuration on the fly, it is advisable to restart the httpd server.)
SessionIdCookieName_str	Name of the cookie used by UIO Apache Proxy to maintain its session (default = OAAM_UIOProxy_SessionId For example, `<Setting name="SessionIdCookieName_str" value="SessionId"/>`
SessionCookie_ DomainLevelCount	Domain level for the UIO Apache Proxy session cookie. Does not affect any other 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 UIO Apache Proxy session cookie. Default is zero which means the expires attribute is not added.
SessionCookie_IsHttpOnly	If set to 1, the UIO Apache Proxy session cookie is marked as HTTP only in the `Set-Cookie` Header. Affects only this cookie. Default is not to mark the cookie as HTTP only. On a supported browser, a `HttpOnly` cookie is only used when transmitting HTTP (or HTTPS) requests, but the cookie value is not available to client side script, hence mitigate the threat of cookie theft via Cross-site scripting.
SessionCookie_IsSecure	If set to 1, UIO Apache Proxy session cookie is marked as secure in the Set-Cookie header. It does not affect any other cookie. The default is not to mark the cookie as secure. A secure cookie is only used when a browser is visiting a server via HTTPS, that will make sure that cookie is always encrypted when transmitting from client to server, and therefore less likely to be exposed to cookie theft via eavesdropping.

Table 6–11 (Cont.) OAAM UIO Proxy Settings.

Flags	Description
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 that can be processed by the proxy and request body bigger than this value will be truncated. This is necessary when the application has POSTs with big files getting uploaded. For example, `<Setting name="MaxReqBodyBytes" value="10240"/>`
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. You can 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.
Setting name	Enables internal profiling for various operations such as per interception phase and prints that out in the logs in microseconds. It should be used only for debugging and profiling in non-production environments as this may impact performance. The logs appear at `INFO` level and also at `TRACE` level

6.3.6.2 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`.

Available UIO Apache Proxy Log4j loggers are listed below.

Table 6–12 UIO Apache Proxy Log4j Loggers

Loggers	Description
config.reader	The UIO_Config XML file loading related classes use this logger.
settings.reader	The UIO_Settings XML file loading classes use this logger.
config.datastore	The UIO_Config XML file loading related classes use this logger.

Table 6–12 (Cont.) UIO Apache Proxy Log4j Loggers

Loggers	Description
config	The UIO_Config XML file loading related classes use this logger.
config.reader.populator	The UIO_Config XML file loading related classes use this logger.
condition	All conditions defined in UIO_Config.xml use this logger.
filter	All filters defined in UIO_Config.xml use this logger.
action	All actions defined in UIO_Config.xml use this logger.
interceptor	All actions defined in UIO_Config.xml use this logger.
requestcontext	HTTP request processing is performed by classes that use this logger.
proxy	HTTP request processing is performed by classes that use this logger.
htmlpage	HTML page related processing is performed by classes that use this logger.
httpreqimpl	HTTP request processing is performed by classes that use this logger.
container	HTTP request processing is performed by classes that use this logger.
sessions	UIO Proxy session management related classes use this logger.
http	Logger that is used to log all HTTP traffic when CaptureTraffic setting is turned on.
distsessions	UIO Proxy session management related classes use this logger.

Note: The logger documentation is provided for completeness and to enable the deployment engineer to make better sense of the logs. Typically for a debugging scenario turn on the log level to `DEBUG` and do not try to filter by any loggers.

6.3.6.3 Application configuration XMLs

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

6.4 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*.

6.5 Setting Up Policies

To set up policies for the UIO Proxy, 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*.

6.6 Configuring the UIO Proxy

The proxy intercepts all HTTP traffic between the client browser and the Web application and performs actions specified in the configuration files. The UIO ISA Proxy uses the XML Schema Definition which is described in the `BharosaProxy.xsd` and the UIO Apache Proxy uses the XML Relax NG definition which is in the `UIO_Config.rng` file in the proxy distribution.

6.6.1 Elements of the UIO Proxy Configuration File

The following sections describe various elements of the proxy configuration file.

6.6.1.1 Components of Interceptors

Interceptors are the most important elements in the proxy configuration. Authoring the proxy configuration file deals mostly with defining interceptors.

There are two types of interceptors: request interceptors and response interceptors. As the names suggest, request interceptors are used when the proxy receives HTTP requests from the client browser and response interceptors are used when the proxy receives HTTP response from the server, for example, 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. The URLs must be an exact match; there is no support for regular expressions. For a request interceptor, this is the set of URLs for which the request interceptor will be executed in the request portion of the HTTP request, for example, on the way from the client to the server. For a response interceptor, the URL is that of the HTTP request; the response interceptor will be executed in the response portion of the HTTP request, for example, while getting the response from the server to the client. If the URL has query parameters, then they should not be listed. You can use conditions to check for any query parameters.
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 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 the following:
    a. 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

Table 6–13 Components of Interceptors

Interceptor	Attributes	Description
RequestInterceptor	id, desc, post-exec-action, isGlobal, enabled	`RequestInterceptor` defines an interceptor that will be run during the request phase. It has an `id`, `description`. Optionally it has a `post-exec-action` that takes the values `continue`, `stop-intercept`, `stop-phase-intercept`; the default is `continue`. Optionally it has `isGlobal` which takes the values `true` or `false` and is `false` by default. It also takes the `enabled` attribute which is also optional and is `true` by default.
ResponseInterceptor	id, desc, post-exec-action, isGlobal, enabled	`ResponseInterceptor` defines an interceptor that is run during the response phase of the HTTP request. The attributes of this element are similar to that of `RequestInterceptor`. This element contains zero or more `RequestUrl` elements, zero or more conditions elements, zero or more filter elements, zero or one target element. The `RequestUrl` element has a single URL element for which this interceptor will execute. The URl must be an exact match. There is no regular expression or pattern support for the URL. Instead of the `RequestUrl` element there is a `ResponseUrl` element which has similar meaning. All else is similar to the `RequestInterceptor`.

6.6.1.2 Conditions

Conditions are used in the 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 6–14 lists conditions that can be defined in an interceptor.

Table 6–14 Conditions Defined in an Interceptor

Condition name	Attributes	Description
HeaderPresent	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	enabled, name	Checks the presence of the specified parameter in the request. Example: `<ParamPresent name="loginID"/>`
QueryParamPresent	enabled, name	Checks the presence of the specified query parameter in the URL. Example: `<QueryParamPresent name="TraceID"/>`

Table 6–14 (Cont.) Conditions Defined in an Interceptor

Condition name	Attributes	Description
VariablePresent	enabled, name	Checks whether the specified proxy variable has been set. Example: `<VariablePresent name="$userid"/>`
RequestCookiePresent	enabled, name	Checks the presence of the specified cookie in the request Example: `<RequestCookiePresent name="SESSIONID"/>`
ResponseCookiePresent	enabled, name	Checks the presence of the specified cookie in the response Example: `<ResponseCookiePresent name="MCWUSER"/>`

HeaderValue	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	enabled, name, value, mode, ignore-case	Checks whether the specified request parameter value matches the given value.
		Example:
		`<ParamValue name="cancel"`
		`value="Cancel"/>`

QueryParamValue	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	enabled, name, value, mode, ignore-case	Checks whether the specified proxy variable value matches the given value.
		Example:
		`<VariableValue name="%REQUEST_METHOD"`
		`value="post"/>`

RequestCookieValue	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"/>`

Table 6–14 (Cont.) Conditions Defined in an Interceptor

Condition name	Attributes	Description
ResponseCookieValue	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	enabled, status	Checks whether the status code of the response matches the given value. Example: `<HttpStatus status="302"/>`

HtmlElementPresent	enabled, name,
	attrib-name1, attrib-value1,
	attrib-name2, attrib-value2,
	…
	attrib-name9, attrib-value9,
		Checks presence of a html element to match the specified conditions:
		`<name attrib-name1="attrib-value1" attrib-name2="attrib-value2" …/>`
		Example:
		`<HtmlElementPresent name="form" attrib-name1="name" attrib-value1="signon"/>`

PageContainsText	enabled, text	Checks whether the response contains the given text. Example: `<PageContainsText text="You have entered an invalid Login Id"/>`
NotVariableValue	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 6–14 (Cont.) Conditions Defined in an Interceptor

Condition name	Attributes	Description

And	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	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	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`.

6.6.1.3 Filters

Filters are used in the 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 6–15 lists filters that can be defined in an interceptor.

Table 6–15 Filters Defined in an Interceptor

Filter name	Attributes	Description
AddHeader	enabled, name, value	Adds the specified header with a given value to request/response. The header name should be terminated by a colon (":"). Example: `<AddHeader name="userid:" value="$userid"/>`
SaveHeader	enabled, name, variable	Saves the specified request/response header value in the given proxy variable. The header name should be terminated by a colon (":"). Example: `<SaveHeader name="userid:" variable="$userid"/>`
RemoveHeader	enabled, name	Removes the specified header from request/response. The header name should be terminated by a colon (":"). Example: `<RemoveHeader name="InternalHeader:"/>`
AddParam	enabled, name, value	Adds a request parameter with a specified name and value. Example: `<AddParam name="loginID" value="$userid"/>`
SaveParam	enabled, name, variable	Saves the specified request parameter value in to the given proxy variable. Example: `<SaveParam name="loginID" variable="$userid"/>`
AddRequestCookie	enabled, name, value	Adds the specified cookie with a given value to request Example: `<AddRequestCookie name="JSESSIONID" value="$JSESSIONID"/>`
SaveRequestCookie	enabled, name	Saves the specified request cookie value in the given proxy variable
AddResponseCookie	enabled, name	Adds the specified cookie with a given value to response Example: `<AddResponseCookie name="JSESSIONID" value="$JSESSIONID"/>`
SaveResponseCookie	enabled, name	Saves the specified response cookie value in the given proxy variable. Example: `<SaveResponseCookie name="JSESSIONID" variable="$JSESSIONID"/>`

Table 6–15 (Cont.) Filters Defined in an Interceptor

Filter name	Attributes	Description

SaveHiddenFields	enabled, form, variable, save-submit-fields	Saves all the hidden, submit fields value, in the given form if the form name is specified to the given proxy variable. To not save submit fields, set save-submit-fields attribute to false.
		Example:
		`<SaveHiddenFields form="pageForm"`
		`variable="%lg_HiddenParams"/>`

AddHiddenFieldsParams	enabled, variable	Adds request parameters for each hidden field saved in the variable.
		Example:
		`<AddHiddenFieldsParams`
		`variable="%lg_HiddenParams"/>`

SetVariable	enabled, name, value	Sets the proxy variable with the given name to the specified value.
		Example:
		`<SetVariable name="$Login-Status"`
		`value="In-Session"/>`

UnsetVariable	enabled, name	Removes the proxy variable with the given name. Example: `<UnsetVariable name="$Login-Status"/>`
ClearSession	enabled, name	Removes all session variables in the current session. Example: `<ClearSession/>`
SaveQueryParam	enabled, name, variable	Saves the specified query parameter in the given proxy variable. Example: `<SaveQueryParam name="search" variable="$search"/>`
SaveRequest	enabled, variable	Saves the entire request content in the given proxy variable. This includes all headers and the body, if present. Example: `<SaveRequest variable="$billPayRequest"/>`
SaveResponse	enabled, variable	Saves the entire response content in the given proxy variable. This includes all headers and body, if present. Example: `<SaveResponse variable="$BillPay-Response"/>`

Table 6–15 (Cont.) Filters Defined in an Interceptor

Filter name	Attributes	Description

ReplaceText	enabled, find, replace	Updates the response by replacing the text specified in `find` attribute with the value given in `replace` attribute.
		Example:
		`<ReplaceText find="string-to-find"`
		`replace="string-to-replace"/>`

ProcessString	enabled, source, find, action, count, search-str, start-tag, end-tag, ignore-case, replace, encoding	This filter can be used to extract a sub-string from a string (such as request, response contents) and save it to a proxy variable. This filter can also be used to dynamically format strings. The `find` attribute has two values: `string` and `sub-string`. It defines the `find` mode as applying to the entire `string` or to `sub-string`. The `sub-string` is defined by the `start-tag` and `end-tag`. If the `find` value is `sub-string`, then only `start-tag` and `end-tag` values are used; otherwise, they are ignored. The action attribute has 3 values: `extract`, `replace` and `eval`. The value of '`extract`' means it will copy the content bracketed by `start-tag` and `end-tag` over to the variable. The value of `replace` is used to perform a `find` and `replace` operation. `eval` is used to `find` and `evaluate` the variable in line. The attribute encoding is optional and can take a value of base64 if you want the resulting string to be base64 encoded. This attribute is supported only on UIO Apache Proxy. See the following examples in Section 6.6.1.4, "Filter Examples - ProcessString" on how to use this filter.
FormatString	enabled, variable, format-str, encoder, param-0, param-1, …, param-n	This filter provides functionality similar to the `sprintf()` C library function: to store a formatted string in a variable. Optionally, the string stored in the variable can be encoded in base64 format. Refer to the example in Section 6.6.1.5, "Filter Examples - FormatString" on using this filter to create a HTTP Basic Authentication header. FormatString is not supported in the UIO Apache Proxy. As it ProcessString provides all the required functionality.

6.6.1.4 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. The action of '`extract`' will extract the first matching `start-tag` and `end-tag` pair.

```
<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. You can also use the count attribute to specify behavior in case there are multiple matches. The attribute '`count`' can take values `all`, `once` or a `number`.

```
<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. You can use the attribute count to specify the behavior in case of multiple matches. This attribute can take the value of '`all`', '`once`' or a `number`.

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

You can specify the attribute `ignore-case` as `true` or `false` and it can be applied to any of the above examples and accordingly the search operation will be case sensitive or not. You can specify encoding attribute optionally and it will encode the resulting string before storing in to the variable. This attribute can take only base64 value. If you do not specify this attribute then the resulting string is stored as is.

The encoding attribute is supported only on UIO Apache Proxy. On UIO ISA Proxy you will have to use FormatString if you want to encode the result in base 64.

6.6.1.5 Filter Examples - FormatString

Here is an example to create a HTTP Basic Authentication response header in variable `$AuthHeader`Value, using the user name/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"/>
```

6.6.1.6 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. It takes has 2 attributes: `url` which contains the URL to which the proxy should re-direct the user and `display-url` which is optional.

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 perform a `HTTP-GET` on the server to get the URL specified in the `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. It has two attributes: `html` which contains the variable that has the saved content that you want send back to the user and optional attribute `display-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 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. It has two attributes: `url` which is the URL to perform the get on and the `display-url` which is optional.

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 perform a `HTTP-GET` on the server to get the URL specified in the `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. It has two attributes: `url` that has the URL to which the post needs to be sent and optional `display-url`.

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 perform a `HTTP-POST` to the server to the URL specified in the `url` attribute.

send-to-server

In certain situations the request from the 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. It has two attributes: `html` which contains the variable that has the saved content and the optional `display-url` attribute.

If the `display-url` attribute is specified in the interceptor, then the proxy will send out a `HTTP 302` redirect response to the browser. This will cause the browser to request for the `display-url` and then the proxy will send out the saved request to the server. If you use this action in a response interceptor, then `display-url` is mandatory; without this, the action will fail.

6.6.1.7 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 life span 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, and other filters.

All types of variables can be unset/deleted by the UnsetVariable filter. The 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 are 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. In the UIO ISA Proxy, global variables can also be set by adding registry values under key HKLM\Software\Bharosa\Proxy\Globals. The 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.

Pre-defined variables

The UIO Proxy supports the following pre-defined request variables:

Table 6–16 Pre-defined Variables Supported by the UIO Proxy

Variable name	Description
%RESPONSE_CONTENT	This variable contains the contents of the entire response from the Web server for the current request. Forthe UIO Apache Proxy, %RESPONSE_CONTENT has been deprecated. Please use SaveResponse, SaveHeader, SaveResponseCookie, and ReplaceText filters instead.
%REQUEST_CONTENT	This variable contains the contents of the entire request from the client. For the UIO Apache Proxy, %REQUEST_CONTENT has been deprecated. You can use SaveRequest, SaveHeader, and SaveRequestCookie filters instead.
%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, and so on.
%REMOTE_HOST	Hostname of the client or agent of the client. (For the UIO Apache Proxy, you need to enable the hostname lookup by using the Apache directive 'HostnameLookups On'.)

Table 6–16 (Cont.) Pre-defined Variables Supported by the UIO Proxy

Variable name	Description
%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

6.6.1.8 Application

A single proxy installation can be used to provide multifactor authentication for multiple Web application that run in one or more Web servers. In the UIO 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 proxy configuration file. These interceptors are called "global" interceptors and will be evaluated and executed prior to the interceptors defined in the applications.

6.6.2 Interception Process

An HTTP messages consist of requests from the client to server and responses from the server to client. HTTP is transaction oriented. A request from client to server will have a single response from the server to client. The request has a set of headers followed by, optionally, a request body. Similarly the response has headers and, optionally, a body. Since the proxy is sitting in between the client and the target application, it can modify the request headers, body and response headers and body of any HTTP request, using the configuration XML. Note that a response could be a normal 200 OK response or it could be a redirect response 302 or any other HTTP status response. In all these cases, the response is for that request and will trigger the response interceptors for the same request. An example, if the request is for the URL `/doLogin.do`, and the response is a `redirect (302)` with the location of `/loginPage.jsp` then all the request and response interceptors will be triggered for the URL `/doLogin.do`. The next HTTP request is a `HTTP GET` on `/loginPage.jsp` and this will cause all the request and response interceptors for `/loginPage.jsp` to be triggered.

When a request arrives, the 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 proxy evaluates response interceptors defined for the URL of the HTTP request in the order defined in the configuration file.

If the conditions in an interceptor evaluate to `true`, the proxy will execute that interceptor i.e. execute the filters and action. After executing an interceptor, the 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`

It is highly recommended that the `post-exec-action` attribute be specified for interceptors that do not 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`. The `stop-phase-intercept` value of `post-exec-action` on a request interceptor stops the request interception but continues with response interception while `stop-intercept` stops the interception completely for that request. For non-global interceptors, the default value is `continue` if no action is specified and `stop-phase-intercept` if an action is specified.

As mentioned earlier the 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.

6.6.3 Configuring Redirection to the Oracle Adaptive Access Manager Server Interface

The UIO Proxy redirects the user to OAAM Server pages at appropriate times, for example to collect the password using OAAM Server, to run risk rules. HTTP headers are used to exchange data between the UIO 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 6–17 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, and elements).
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 To put an URL with an "`&`" into a target action so that the xml parser does not have an error, you must escape it: `&`
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.

Table 6–17 (Cont.) OAAM Server Interface

URL	Condition	Action
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 the "`OAAM ServerPhase`" header to "`one`". A "`?`" is accepted in a URL specified in a target action. In a target action URL, you would have the "`?`" and any parameters after it Setting "`Login-Status`" to ▪ `success` will update the session status for the user in OAAM to `success` and run post-authentication rules. ▪ `wrong_password`, `invalid_user`, `user_disabled`, `system_error` will update the session status in OAAM to the status passed and the user will be taken to the login page with the appropriate error messaging
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` Setting "`Login-Status`" to ▪ `success` will update the session status for the user in Oracle Adaptive Access Manager to success and run post-authentication rules. ▪ `wrong_password`, `invalid_user`, `user_disabled`, `system_error` will update session status in Oracle Adaptive Access Manager to the status passed and the user will be taken to the login page with appropriate error messaging
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.

Table 6–17 (Cont.) OAAM Server Interface

URL	Condition	Action
registerQuestions.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 log off 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`
		In most cases control is not given to the proxy via a response header in a block situation. Instead, the user is taken to the following URL with a query parameter "`action`" set to the error code "`block`". This presents the user with the OAAM Server login page with a message stating the reason they are there.
		`/error.do?action=block`
		Alternatively it is possible to get the same result with the following URLs.
		`/loginFail.do?action=block`
		`/loginPage.jsp?action=block`

logout.do	On completion of application session logout	Redirect to this URL to log out the OAAM Server session
logout.do	On receiving response	Redirect to application logout URL to log off the application session, if it is not already off
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
changeUserId.do	On receiving response	Redirect to the appropriate application page or send back the 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`.

Table 6–17 (Cont.) OAAM Server Interface

URL	Condition	Action
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 to the forgot-password flow 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 a 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>"

6.7 Application Discovery

Two flags in the settings are used for application discovery. One flag instructs the proxy to ignore its configuration XML and act as a reverse-proxy only. The other flag instructs the proxy to capture all the HTTP traffic and print it to the logs. The first flag is used for application discovery to capture the HTTP traffic and analyze it. The second flag would be kept on during the configuration XML development phase to debug the configuration XML itself.

Application discovery is the process of studying an existing Web application to author the proxy configuration to add multifactor authentication using the UIO Proxy. A 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 then be analyzed to author the proxy configuration. The UIO Proxy should be set up to dump all the HTTP traffic to a file. Then a few logins/login attempts to the application should be made via the proxy. The captured HTTP traffic should then be analyzed to author the proxy configuration.

6.7.1 Application Information

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

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

1. URL to log in to the application.
2. Test user account credentials, including the data required in the forgot password scenario. It will be best to get as many test accounts as possible, preferably at least five accounts, for uninterrupted discovery and testing. Note that during the discovery process some accounts could become disabled, due to multiple invalid login attempts.

3. Contact (phone, email) to enable/reset test accounts

6.7.2 Setting Up the UIO ISA Proxy

The Microsoft ISA server should be set up to publish the Web application under discovery, for example, 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 UIO ISA Proxy. Refer to the Microsoft ISA configuration document for details of setting up Microsoft ISA.

The UIO ISA Proxy settings (registry values under `HKLM\SOFTWARE\Bharosa\Proxy` key) should be set as given in Table 6–18 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 6–18 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 (for example, successful login attempt, wrong password, wrong user name, disabled user, and other scenarios) in separate files. `TraceFilename` 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 given in Table 6–19 to restore the default UIO ISA Proxy behavior.

Table 6–19 Proxy settings after application discovery

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

6.7.3 Setting Up the UIO Apache Proxy

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

Table 6–20 shows the settings (in `UIO_Settings.xml`) to enable this mode of operation.

Table 6–20 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 user name, 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 6–21, to restore the default UIO Apache Proxy behavior.

Table 6–21 Settings to restore default proxy behavior

Settings	Value
IgnoreUrlMappings	0
CaptureTraffic	0

6.7.4 Scenarios

Collect information for the following scenarios during the discovery process.

You must create interceptors in the `TestConfig.xml` file that look for certain URLs and conditions in the HTTP traffic. The proxy listens to the HTTP traffic and when it sees a URL that matches a URL in its `TestConfig.xml` file, it evaluates the interceptors that have a URL match and it evaluates the conditions block in the interceptor. If they match, the UIO Proxy executes the filter block and condition block.

Login

1. URL that starts the login process
2. URL that contains the login form
3. Names of the input fields like user name, 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, and so on)
   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 the 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

Follows 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, the `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 must be reviewed for understanding. Most applications ask for alternate ways to identity the user (account number/PIN, SSN/PIN, question/answer, and other ways); some applications provide more than one option. Some applications let the user reset the password after 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.

6.8 Samples

The proxy configuration to add multifactor authentication to the BigBank Web application is shown below. The BigBank web application is a sample application which shows a login flow. The example will demonstrate the integration of the UIO Proxy into the login flow of an application.

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/">

  <RequestInterceptor id="AddAppIdToOAAMServerRequests-BigBank"
```

```
                      desc="Add BharosaAppId header to each request to oaam_
server"
                      post-exec-action="continue">
    <Conditions>
      <VariableValue name="%URL"
                     value="/oaam_server/"
                     mode="begins-with"
                     ignore-case="true"/>
    </Conditions>

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

    <SetGlobal name="@Phase1Enabled" value="false"/>
    <SetGlobal name="@Phase2Only"    value="true"/>

  <Application id="BigBank">

    <!-- In phase one, we use BigBank's login form to collect username/password
-->
    <!-- In phase two, we use oaam_server login forms to collect username and
password -->

    <!-- Disable this interceptor after phase one is retired -->
    <RequestInterceptor id="Phase1BigBankLoginPostRequest"
                        desc="get the loginid from the post parameters"
                        post-exec-action="continue" enabled="@Phase1Enabled">
      <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 Bharosa
login page"
                        enabled="@Phase2Only">
      <RequestUrl url="/bigbank"/>
      <RequestUrl url="/bigbank/"/>
      <RequestUrl url="/bigbank/loginPage.jsp"/>

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

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

      <Conditions>
```

```
        <VariableValue name="%REQUEST_METHOD" value="POST"/>
        <ParamPresent  name="userid"/>
        <Not>
          <ParamPresent name="password"/>
        </Not>
      </Conditions>

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

    <ResponseInterceptor id="Phase2PasswordPageResponse"
                         desc="Save the userid, decoded password from Bharosa
Password Page response">
      <ResponseUrl url="/oaam_server/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/loginPage.jsp"
              display-url="/bigbank/GetLoginPage"/>
    </ResponseInterceptor>

    <ResponseInterceptor id="GetBigBankLoginPageResponse"
                         desc="Save values of all 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"/>
      </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"/>
```

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

      <Target action="redirect-client" url="/oaam_server/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="/oaam_server/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="$Submit-Credentials-Response"/>
        <AddHeader name="Login-Status:" value="$Login-Credentials-Status"/>
      </Filters>

      <!-- Target action="redirect-client" url="/oaam_
server/UpdateLoginStatusPage"/ -->
      <Target action="get-server" url="/oaam_server/updateLoginStatus.do"/>
    </ResponseInterceptor>

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

      <!-- Any interceptors for /bigbank/login.do will not run because we are
```

```
doing get-server. -->
      <Target action="get-server" url="/oaam_server/login.do"/>
    </RequestInterceptor>

    <RequestInterceptor id="Phase2UpdateLoginStatusPageRequest"
                        desc="Update Bharosa Tracker with the login status">
<!--post-exec-action="continue" -->
      <RequestUrl url="/oaam_server/UpdateLoginStatusPage"/>

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

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

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

```
      <ResponseUrl url="/oaam_server/updateLoginStatus.do"/>
      <ResponseUrl url="/oaam_server/challengeUser.do"/>
      <ResponseUrl url="/oaam_server/registerQuestions.do"/>
      <ResponseUrl url="/oaam_server/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="/oaam_
server/loginPage.jsp?action=invalid_user"/>
    </ResponseInterceptor>

    <ResponseInterceptor id="BlockLoginResponse"
                         desc="BigBank passed login but tracker returned 'block' -
fail the login">
      <ResponseUrl url="/oaam_server/UpdateLoginStatusPage"/>
      <ResponseUrl url="/oaam_server/updateLoginStatus.do"/>
      <ResponseUrl url="/oaam_server/challengeUser.do"/>
      <ResponseUrl url="/oaam_server/registerQuestions.do"/>
      <ResponseUrl url="/oaam_server/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 login; 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="/oaam_
server/loginFail.do?action=block"/>
    </ResponseInterceptor>

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

      <Filters>
        <ClearSession/>
      </Filters>

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

    <ResponseInterceptor id="LogoutPageResponse"
                         desc="Bharosa logout selected; logoff the session in
BigBank">
      <ResponseUrl url="/oaam_server/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="/oaam_server/loginPage.jsp"/ -->
    </ResponseInterceptor>

  </Application>
</BharosaProxyConfig>
```

6.8.1 Descriptions for Interceptors

Descriptions of the various interceptors that are defined in the sample configuration are summarized in Table 6–22.

Table 6–22 Sample Configuration Interceptors

Interceptor ID	Type	Explanation
AddAppIdTobharosauioRequests-BigBank	Request	Set headers for all requests for OAAM Server. Invoked by any request to OAAM Server.
Phase1BigBankLoginPostRequest	Request	Get login ID from post parameters, set Phase One, save user ID. Invoked by request for `/bigbank/login.do` when `Phase one` is enabled.
Phase2RedirectBigBankLoginPageRequest	Request	Redirect login page from application to OAAM Server. Invoked when Phase Two is enabled and application login page is requested
Phase2BharosaLoginPageRequest	Request	Set Phase Two and save variables. Invoked by request for OAAM Server `login.do`
Phase2PasswordPageResponse	Response	Save ID/Password in header, redirect client to BigBank loginpage. Invoked by response from OAAM Server's `password.do`.
GetBigBankLoginPageResponse	Response	Save all hidden fields values, then post login credential to BigBank. Invoked by response from `/bigbank/GetLoginPage`.
InvalidLoginResponse	Response	Actions to take when getting invalid login response from BigBank.
WrongPasswordResponse	Response	Actions to take when getting wrong password response from BigBank.
LoginSuccessResponse	Response	Actions to take when getting login success response from BigBank.
Phase1UpdateLoginStatusPageRequest	Request	Set Phase One and add headers. Invoked by request for OAAM Server to update status after getting login response from BigBank.
Phase2UpdateLoginStatusPageRequest	Request	Add header and update OAAM Server with login status. Invoked by request for `oaam_server/updateLoginStatusPage`.
AllowLoginResponse	Response	Set variables and direct client to the next page to continue with the login. Invoked when receiving login success response from OAAM Server.
Phase1FailLoginResponse	Response	Set login status and direct client to next page. Invoked in Phase One when BigBank failed the login and the response sent back from OAAM Server.
FailLoginResponse	Response	Set login status and redirect client to theOAAM login block page. Invoked when BigBank failed the login and Phase One is not enabled.
BlockLoginResponse	Response	Set `Block` status and redirect client to BigBank login blocked page. Invoked when BigBank passed login but OAAM Server decided to `block`.
LoginBlockedPageRequest	Request	Redirect client to BigBank logout page. Invoked by request for BigBank Login Blocked page.
Phase1LoginBlockedPageResponse	Response	Clear session and redirect client to the OAAM Login Blocked page, then stop intercept. Used in Phase One, invoked by response from BigBank Login Blocked page.
Phase2LoginBlockedPageResponse	Response	Clear session and redirect client to OAAM Login Blocked page. Used when Phase One is not enabled, invoked by response from BigBank Login Blocked page.

Table 6–22 (Cont.) Sample Configuration Interceptors

Interceptor ID	Type	Explanation
LogoutPageResponse	Response	Redirect client to BigBank logout page. Invoked by response from OAAM logout page.
Phase1LogoffPageResponse	Response	Clear session when getting response from BigBank logout page. Used when Phase One enabled.
Phase2LogoffPageResponse	Response	Clear session when getting response from BigBank logout page. Used when Phase Two enabled.

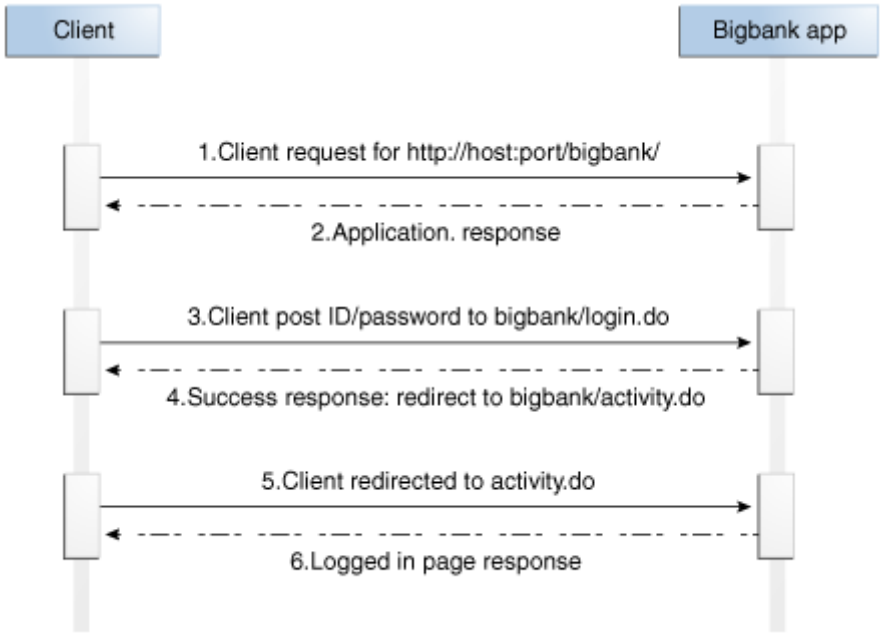
6.8.2 Flow for BigBank without UIO Proxy

The following is the flow of the BigBank application without the UIO Proxy for login and logout.

6.8.2.1 Login

The Login without UIO Proxy flow is shown below.

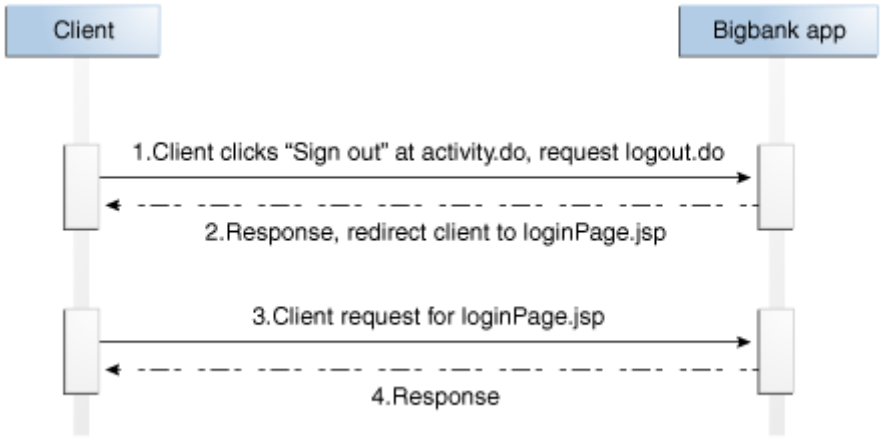
Figure 6–3 Login Flow - Without UIO Proxy



6.8.2.2 Logout

The Logout without UIO Proxy flow is shown below.

Figure 6–4 Logout - Without UIO Proxy



6.8.3 Flow for First-time User to Log In and Log Out of BigBank with UIO Proxy

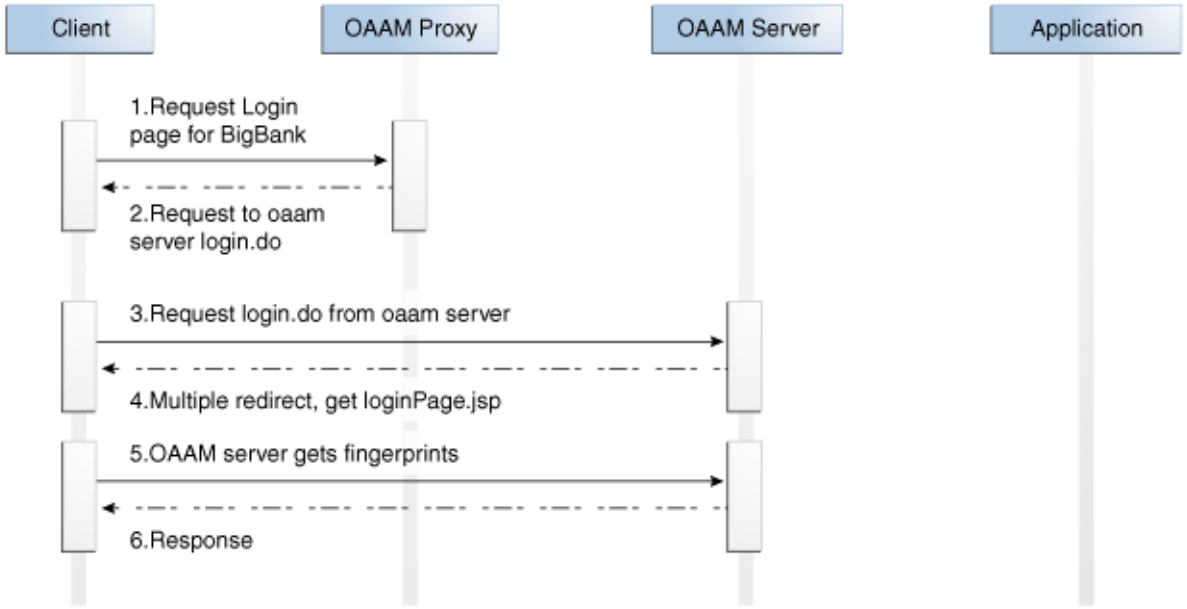
This section provides details for the flows for first time users who log in to the BigBank application through the UIO Proxy. The regular flow, including the login phase, registration phase/skip registration phase, and logout phase, and the deviation flow (block login) are covered. Interceptors defined in Configure xml that are used in each step in the flow will be listed.

Note: For the proxy, the only messages shown are ones whenthe interceptors match request/response. Normal messages that the proxy passes betweenthe client and Oracle Adaptive Access Manager/application are skipped to simplify the scenario.

The regular flow (four phases) consists of the login, registration, skip registration, and logout phases.

Login phase:

Figure 6–5 Flow for Getting Login Page



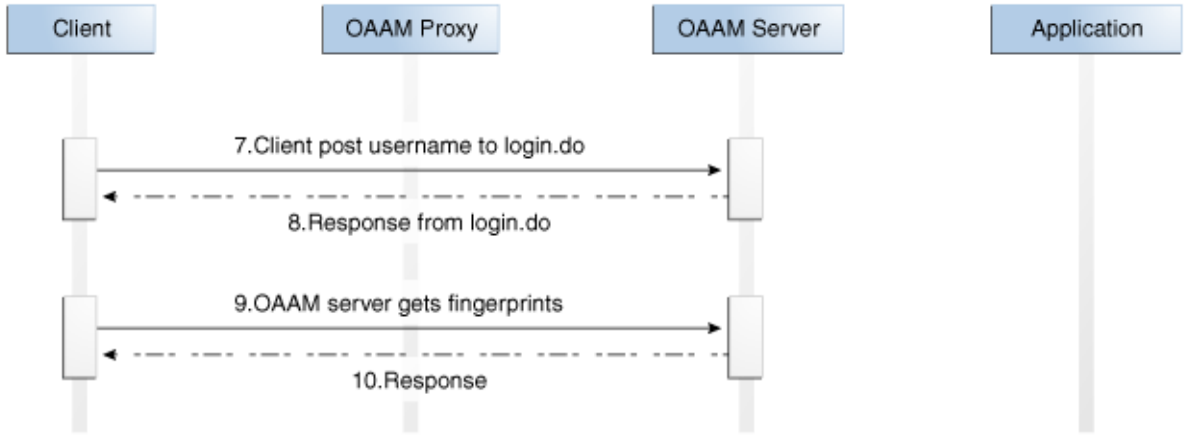
1. Client requests Login page for the application (`http://proxyhost:port/bigbank`).
2. The proxy intercepts the request, and sets the headers. Then, the proxy redirects the client to `oaam_server/login.do`.

   The request is intercepted by two interceptors: `AddAppIdTobharosauioRequests-BigBank` and `Phase2RedirectBigBankLoginPageRequest`.

   Note: `AddAppIdTobharosauioRequests-BigBank` sets the HTTP headers and variables. It will intercept any request for the OAAM Server and the proxy will try other interceptors to see if there are more matches after this interceptor.

   `Phase2RedirectBigBankLoginPageRequest` redirects the client from the BigBank Login page to `oaam_server/login.do`.
3. The client requests to get `login.do` at the OAAM Server (`http://proxyhost:port/oaam_server/login.do`).
4. OAAM Server redirects to Jump page to fingerprint the client device.
5. OAAM Server gets fingerprinting from the client browser.
6. OAAM Server responds after getting the fingerprint with the Login page.

Figure 6–6 OAAM Server responds after getting the fingerprint with the Login page



7. The client posts the user name to the OAAM Server (`http://proxyhost:port/oaam_server/login.do`).

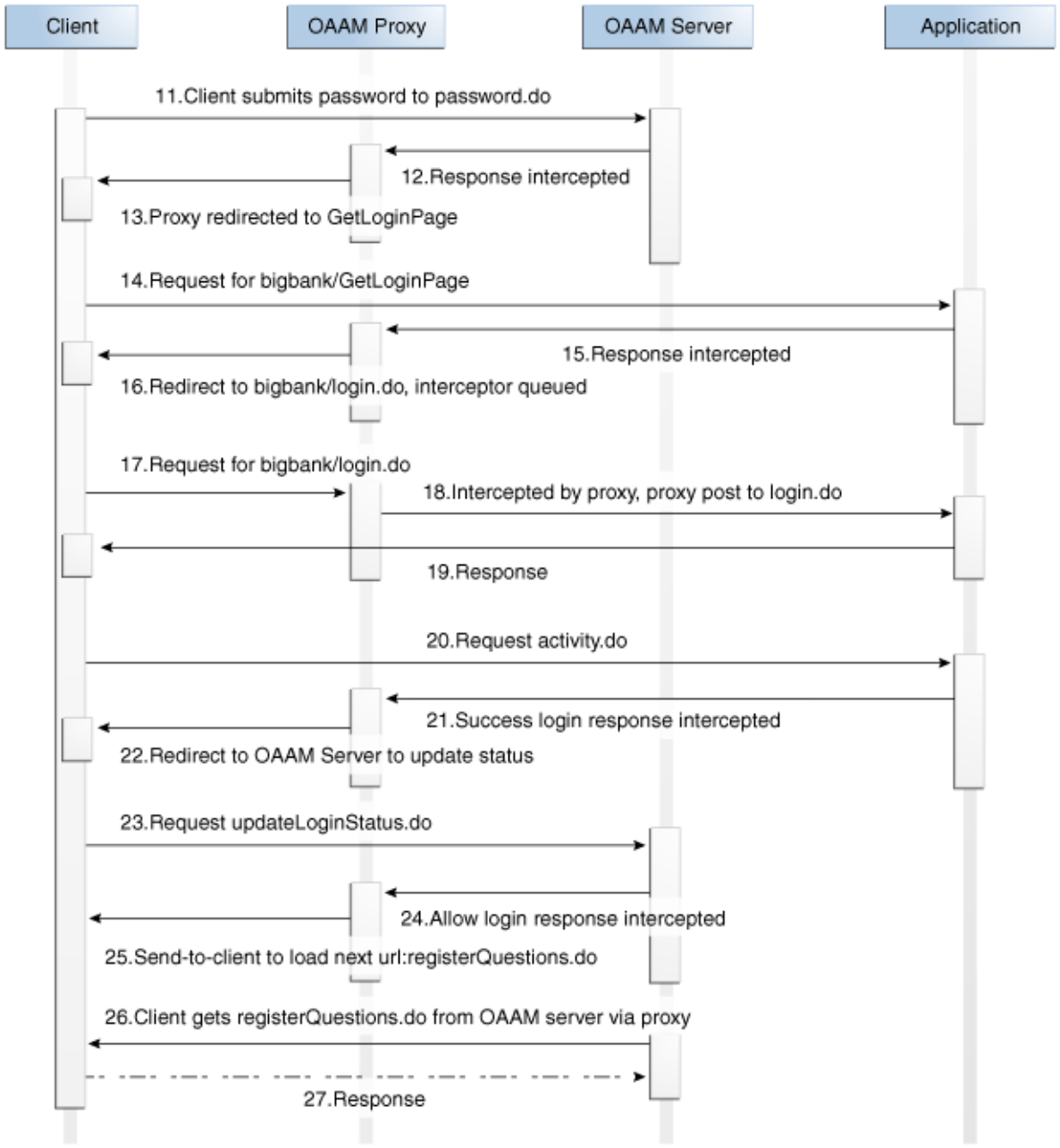
   Other than the `AddAppIdTobharosauioRequests-BigBank` interceptor, the request is intercepted at the proxy by the `Phase2BharosaLoginPageRequest` interceptor. The proxy sets `WebUIOPhase` to two.

8. The OAAM Server responds.

9. The OAAM Server gets fingerprints.

10. The OAAM Server responds after getting fingerprints, with the Password Collection page which has a strong authentication device.

Figure 6–7 Fingerprint and password collection



11. The client submits the password to the OAAM Server (`http://proxyhost:port/oaam_server/password.do`)
12. The OAAM Server responds.

    The response is intercepted by `Phase2PasswordPageResponse`. The proxy saves the headers which contain the Login ID and the password that have been collected by the OAAM Server so far and redirects the client to `/bigbank/GetLoginPage`.
13. The proxy redirects the client to `GetLoginPage`.

14. The client sends a request to BigBank for `GetLoginPage` (`http://proxyhost:port/bigbank/GetLoginPage`).
15. BigBank sends back a response.

    The response is intercepted at the proxy by `GetBigBankLoginPageResponse`. The proxy saves the parameters and performs a `Post-server` action for `/bigbank/login.do`. This is the normal authentication flow for the BigBank application.
16. The proxy queues the interceptor and redirects client to `bigbank/login.do`.
17. The client requests for `login.do` (`http://proxyhost:port/bigbank/login.do`).
18. The request is intercepted by the proxy. The proxy executes the queued interceptor (`GetBigBankLoginPageResponse`) and changes the request method from `GET` to `POST`.
19. BigBank responds, redirect the client to `activity.do`. This is the normal authentication flow for the BigBank application.
20. The client requests for `activity.do` (`http://proxyhost:port/bigbank/activity.do`).
21. BigBank sends a login success response.

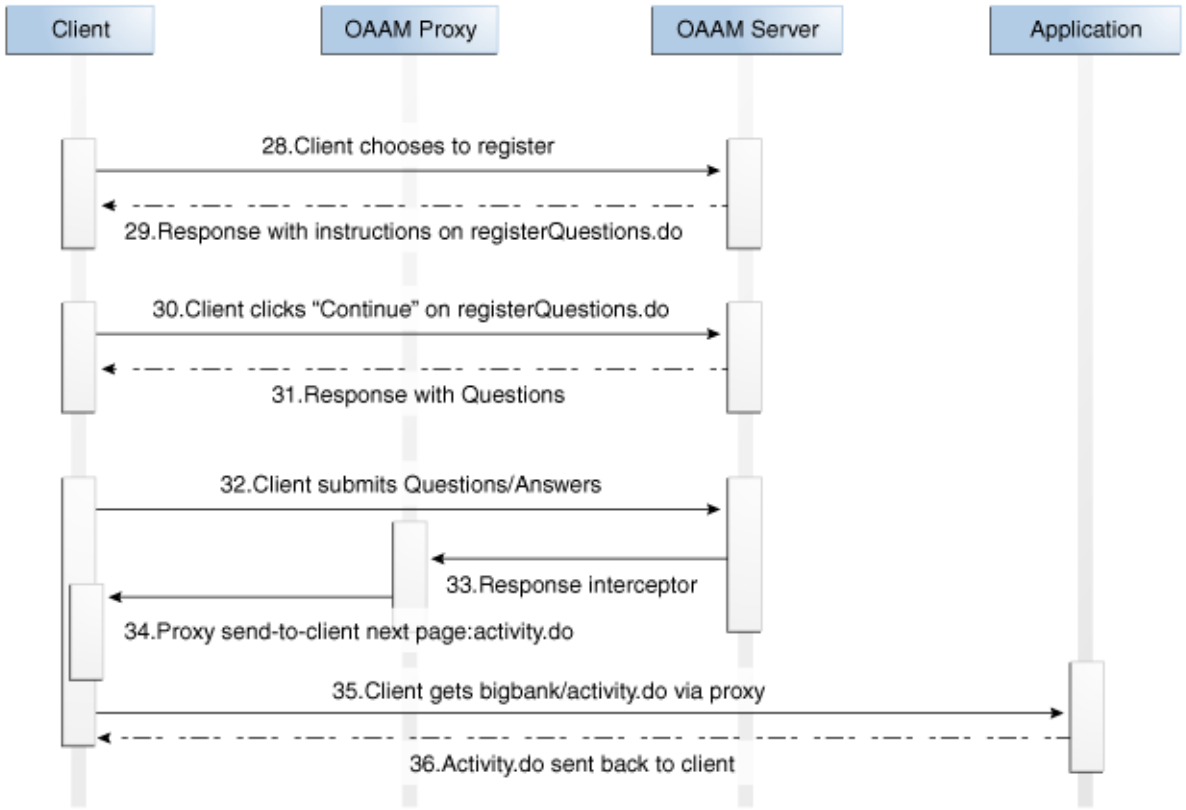
    The response is intercepted at the proxy by `LoginSuccessResponse`. The proxy sets the login status to `success` and performs a `get server` action for `/oaam_server/updateLoginStatus.do`
22. The proxy redirects the client to `updateLoginStatus.do`.
23. The client sends a request to OAAM Server to update the status

    `http://proxyhost:port/oaam_server/updateLoginStatus.do`).
24. OAAM Server does a post authentication check and returns the result.

    The response is intercepted at the proxy by `AllowLoginResponse`.
25. The proxy takes the `send-to-client` action. It sets the `display-url` variable so that the client will request this URL after receiving the response.
26. The client sends a request to OAAM Server for the first-time user to get the Registration page (`http://proxyhost:port/oaam_server/registerQuestions.do`).
27. The Response page has two options for the users: skip and register.

    Registration Flow (client chooses to register):

Figure 6–8 Flow for first-time user to register questions/answers with OAAM Server

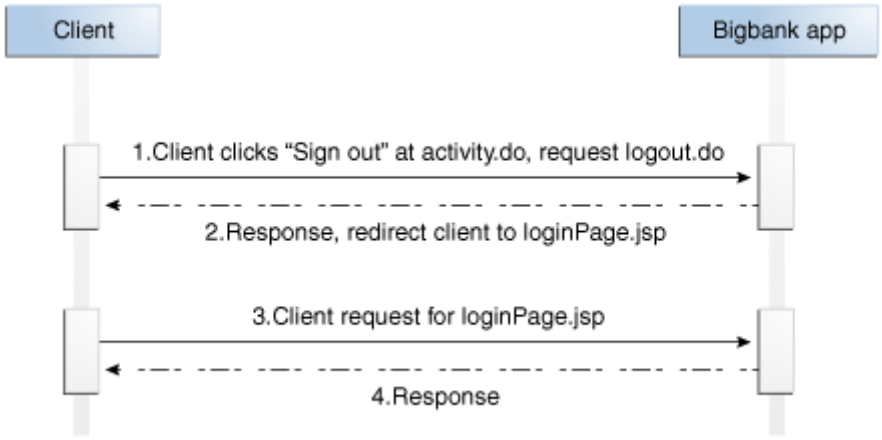


28. The client chooses to register (`Post` to `http://proxyhost:port/oaam_server/registerQuestions.do`).
29. OAAM Server responds with instructions.
30. The client clicks **Continue** on the instruction page. (`http://proxyhost:port/oaam_server/registerQuestions.do`).
31. OAAM Server responds with the Question page.
32. The client selects Questions/Answers and submits them to the OAAM Server (`http://proxyhost:port/oaam_server/registerQuestions.do`).
33. OAAM Server updates the information and responds.
34. The proxy performs a `send-to-client` to the Next page.

    The response is intercepted at the proxy by the `AllowLoginResponse` interceptor. The proxy takes the `sends to Client` action by specifying the Next page after successful authentication. The client will then be redirected to the application page on the next step.
35. The client requests the Next page through the proxy (`http://proxyhost:port/bigbank/activity.do`).
36. The application page (`activity.do`) is sent back to the client through the proxy. This is where the login process ends.

    Logout Phase:

Figure 6–9 Flow for users to log out of BigBank

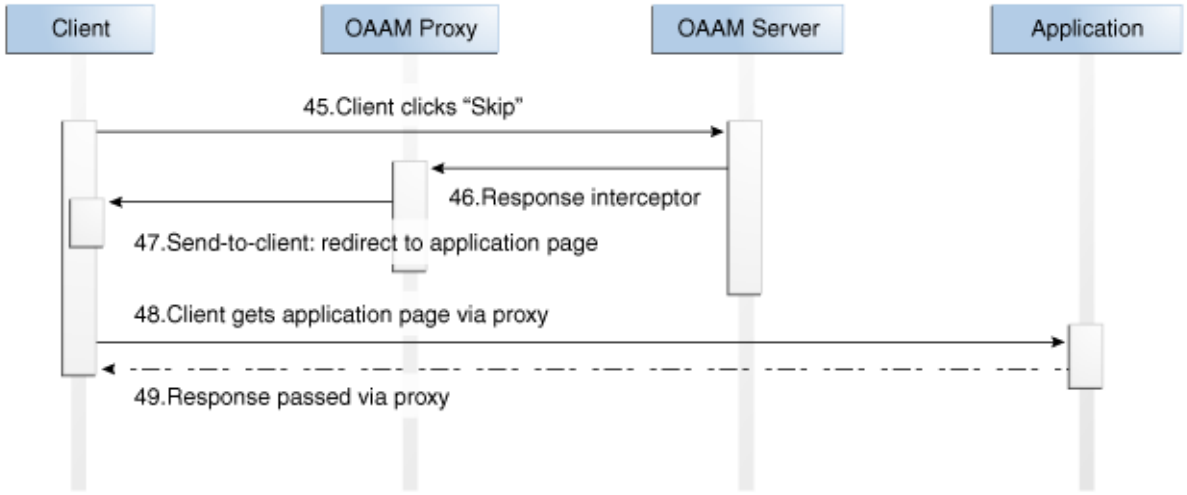


37. The client clicks **Log out** (`http://proxyhost:port/bigbank/logout.do`).
38. The application sends back a response and redirects the client to `bigbank/loginPage.jsp`.

    The response is intercepted by `Phase2LogoffPageResponse`, which clears the session variables.
39. The client requests for the BigBank Login page (`http://proxyhost:port/bigbank/loginPage.jsp`).
40. The proxy intercepts the request and redirects the client to OAAM Server.
41. The client makes a request to OAAM Server for `login.do` (`http://proxyhost:port/oaam_server/login.do`).
42. OAAM Server redirects to the Jump page to fingerprint the client device.
43. OAAM Server fingerprints the client browser.
44. OAAM Server responds after fingerprinting with the Login page.

    Skip Registration phase: Client chooses to skip registration of questions. This phase happens after Login phase in regular flow.

Figure 6–10 Flow occurs after user chooses to skip registration with OAAM Server

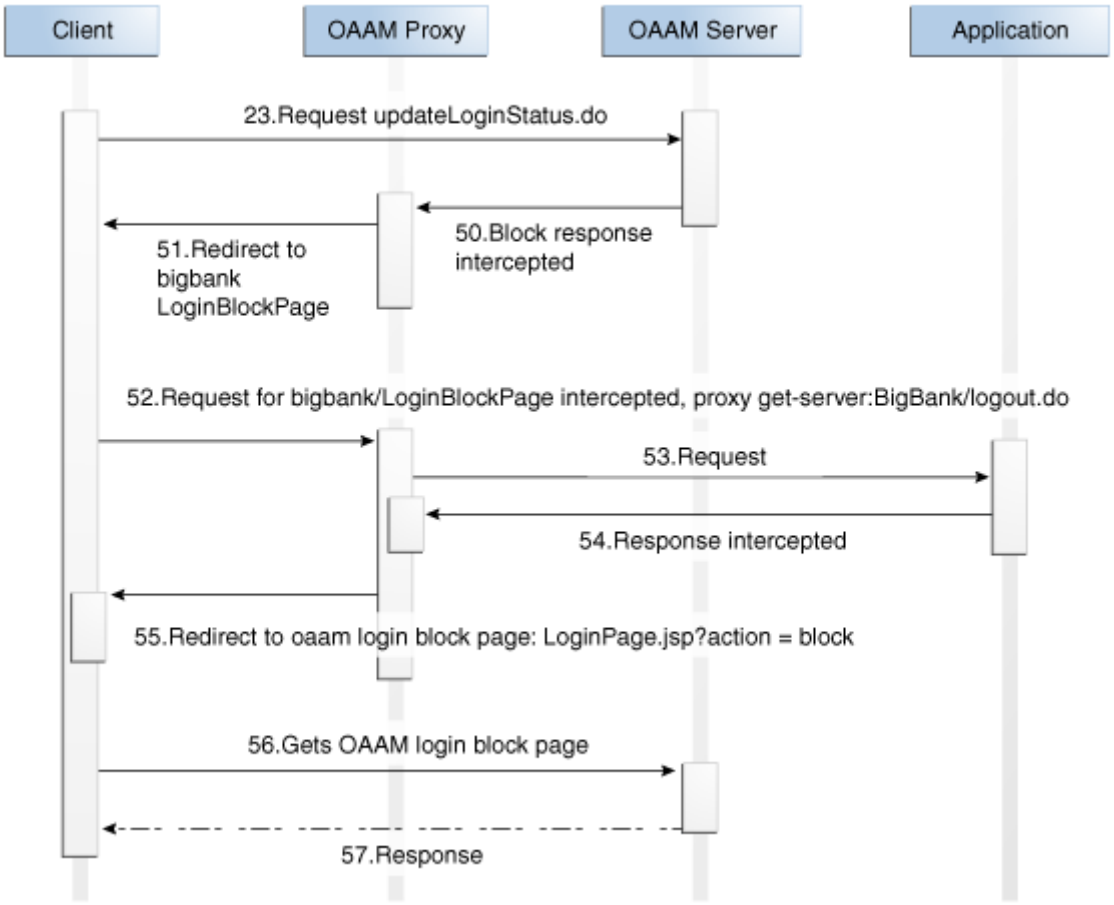


45. The client chooses to skip the registration (Post to `http://proxyhost:port/oaam_server/registerQuestions.do`).
46. OAAM Server responds.
47. The proxy intercepts the response and redirects the client.

    The response is intercepted by `AllowLoginResponse`. The proxy uses `send-to-client` to specify the next step for the client.
48. The client requests for the page specified by the proxy (`http://proxyhost:port/bigbank/activity.do`).
49. The BigBank application sends back a response.

    Deviation flow - Block login: happens when OAAM Server decides to block client after post authentication check. This flow replaces step 15-19 in login phase of regular flow.

Figure 6–11 Deviation flow: user blocked by OAAM Server



50. OAAM Server decides to block the user after post authentication check.

The response is an interceptor by the `BlockLoginResponse` interceptor. This interceptor redirects the client to the application Block page: `/bigbank/BlockLoginPage`

51. The proxy redirects the client to `loginBlockPage` of BigBank.

52. The client requests for BigBank `BlockLoginPage` (`http://proxyhost:port/bigbank/loginPage.jsp?action=block`).

53. The request is intercepted by `LoginBlockedPageRequest` by the proxy. The proxy accepts the `get-server` action for the Logout page: `/bigbank/logout.do`. This action ends the session at BigBank.

54. The application responds.

The response is intercepted by `Phase2LoginBlockedPageResponse`. The proxy clears the session and redirects the client to the OAAM Login Block page.

55. The proxy redirects the client to the OAAM Login Block page.

56. The client requests the Block page from OAAM Server (`http://proxyhost:port/oaam_server/loginPage.jsp?action=block`).

57. OAAM Server responds with Blocked page

6.9 Upgrading the UIO Apache Proxy

Oracle Adaptive Access Manager patches may contain updates for the UIO Apache Proxy for Microsoft Windows and Linux (rhel4). Follow the instructions in this chapter to replace the `mod_uio.so` and related `.dlls` (on MS Windows) and `.so` (on Linux) libraries with those released as part of this patch release.

6.9.1 UIO Apache Proxy Patch Installation Instructions

Installation of a patch is similar to installing the UIO Proxy package. A patch will contain only the modified files. It is good practice to back up all your existing files since the patch will overwrite some or all of the files.

General instructions are given below. A patch contains only the modified files; so if a file is not available in the patch, skip that step. The steps are to be performed manually by the patch installer.

For both MS Windows and Linux:

1. Shut down the instance of Apache that you are updating

   Note: Ensure that you are using `Apache httpd`, version 2.2.8 with `mod_ssl`.

2. Back up existing files: `binary`, `.rng` and `.xml` files
3. Unzip `patch_oaam_win_apache_uio.zip` (for Windows) or `patch_oaam_rhel4_apache_uio.zip` (for Linux), which are located in the `oaam_uio` directory.
4. Copy the binary files from the patch (additionally on Linux, you need to set soft-links to `.so` files appropriately).
5. Copy `UIO_Settings.rng` and `UIO_Config.rng` files from the patch.
6. Compare your existing `UIO_Settings.xml` and `UIO_log4j.xml` files with those given in the patch and verify that you have the correct settings. Refer to the sections that apply to this patch and ensure that you have the correct settings. The same also applies to your configuration `XML` files.
7. Start Apache and run your sanity tests
   - For Windows,
     - The binary files are: `mod_uio.so`, `log4cxx.dll`, `libxml2.dll`, `apr_memcache.dll` (`apr_memcache.dll` was introduced in 10.1.4.5.bp1)
     - The configuration files are: `UIO_Settings.rng`, `UIO_Config.rng`, `UIO_Settings.xml`, `UIO_log4j.xml` and application configuration `XML` files
   - For Linux,
     - The binary files are: `mod_uio.so`, `liblog4cxx.so.0.10.0.0`, `libxml2.so.2.6.32`, `libapr_memcache.so.0.0.1`
     - The binary configuration files are: `UIO_Settings.rng`, `UIO_Config.rng`, `UIO_Settings.xml`, `UIO_log4j.xml` and application configuration `XML` files

6.9.2 UIO Apache Proxy Patch Backout Instructions

Restore the files that you had backed up before you installed the patch.

6.10 Upgrading the UIO ISA Proxy Server

To upgrade the UIO ISA Proxy Server:

1. Stop the Microsoft ISA Server with the following command:

   ```
   net stop fwsrv
   ```

2. Back up the current UIO ISA Proxy Server DLL. The DLL should usually be at: `%ProgramFiles%\Microsoft ISA Server\BharosaProxy.dll`.
3. Overwrite the existing DLL with the one from the patch.
4. Start Microsoft ISA Server with the following command:

   ```
   net start fwsrv
   ```

Part III

Customization and Extensions

Part III contains the following chapters:

7

Customizing Oracle Adaptive Access Manager

The chapter provides information on how to customize Oracle Adaptive Access Manager by using the OAAM Extensions Shared Library.

It contains the following sections:

7.1 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 OAAM Extensions Shared Library.

This shared library, `oracle.oaam.extensions.war`, is located in `<IAM_Home>/oaam/oaam_extensions/generic`. It is deployed in both the OAAM Server and OAAM Admin Server. By default `oracle.oaam.extensions.war` contains the `MANIFEST.MF`, which has the definition of the shared library.

7.2 Add Customizations Using the OAAM 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 ensure it contains the following lines:

```
Extension-Name: oracle.oaam.extensions
Specification-Version:99.9.9.9.9
Implementation-Version:99.9.9.9.9
```

The specification version and implementation version must be more than the versions in the file currently. For example, if the implementation version in the file is 11.1.1.3.0, you could change it to 99.9.9.9.9.

4. Compile custom java classes that extend or implement Oracle Adaptive Access Manager classes, adding the jars from the `$ORACLE_IDM_HOME\oaam\cli\lib` folder to the build class path.
5. Add the custom jars and files as described:
   a. Add the custom jars to the `<IAM_Home>\oaam\oaam_extensions\generic\WEB-INF\lib` folder.
   b. Add custom properties to a file named `bharosa_server.properties` and save it in the `<IAM_Home>\oaam\oaam_extensions\generic\WEB-INF\classes` folder.

      Information about enums are provided in Section 7.3, "User-Defined Enumerations."

   c. Add custom JSPs to the `oaam_extensions` folder.
6. Rejar `oracle.oaam.extensions.war` from the parent folder of `oaam_extensions` using the command:

```
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.

7.3 User-Defined Enumerations

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` directory.

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
```

8

Customizing the OAAM Server

This chapter provides information on customizing the client-facing OAAM Server Web application. The OAAM UIO Proxy offers multifactor authentication to Web applications without requiring any change to the application code. The OAAM Server configuration is specific to the UIO Proxy deployment. Refer to the architectural diagram (Figure 8–1) for the components involved.

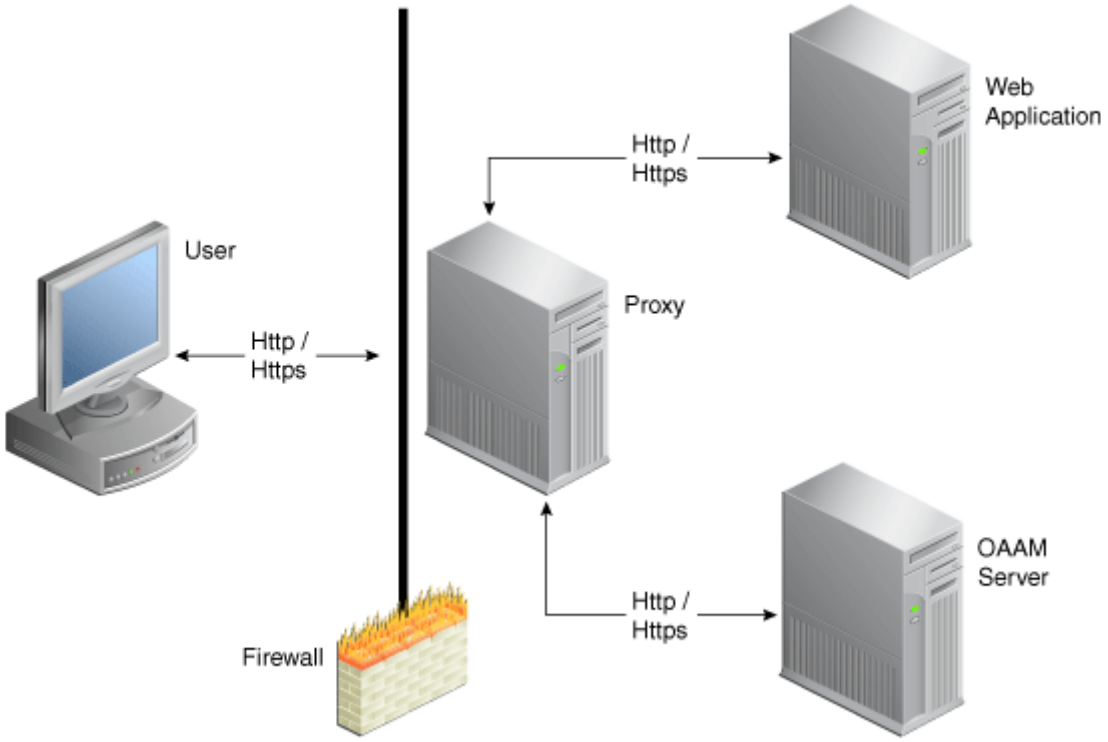
The user interface provided by the OAAM Server Web application can be easily customized to achieve the look and feel of the customer applications. This chapter is intended for integrators who install and configure OAAM Server to support one or more Web application authentication and user registration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

This chapter contains the following sections:

8.1 Architecture

Figure 8–1 shows the UIO Proxy deployment.

Figure 8–1 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.

8.2 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`
  - for client-configured properties (any properties that have been customized for a specific deployment)
  - for UIO Proxy 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 UIO Proxy 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` file is 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 7, "Customizing Oracle Adaptive Access Manager."

8.3 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.

8.3.1 Determining the Application ID

The UIO Proxy 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 6, "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 will be discussed 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=app1Group
```

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

8.3.2 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 appears 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".

8.4 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

8.4.1 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>`, and so on) are already provided by OAAM Server. As a simple example, a header and footer are created 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
        <img src="/client/app1/images/header.jpg" alt="Welcome to App1"/>
```

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

```
/client/app1/footer.jsp
       <img src="/client/app1/images/footer.jsp" alt="App1 Footer"/>
```

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

To associate these files with the application you 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
```

8.4.2 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 8.5, "Configuring Application Properties."

In this example you 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 your newly created file, you will add the following property to bharosa_server.properties:

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

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

8.4.3 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 the example "appId1" application, you would add the following line to `client_resource_<locale>.properties`.

To change the page title on the login page in the example "appId1" application, you 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 you 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.
```

8.5 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.

8.5.1 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
```

8.5.2 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=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.

8.5.3 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 8.5.2, "User-Defined Enums," if you wanted to change "Company ID" to "Profile ID" for only one application (appId1), you would need to modify the enum:

```
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 7, "Customizing Oracle Adaptive Access Manager."

8.5.4 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 8.5.3, "Overriding Existing User-Defined Enums," you would add the following line to remove "Profile ID" from the elements used by the application:

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

9

Customizing User Flow

OAAM supports the customization of user flow. The Struts/Tiles framework is used by OAAM to create a common look and feel for an application.

9.1 OAAM Struts/Tiles Framework

OAAM uses the Struts framework to define the user interface flow. The Struts configuration file (`/WEB-INF/struts-config.xml`) defines all the navigation rules in the form of Struts action definitions. Action definitions typically contain path, type, and parameter attributes. Many definitions also contain one or more forward elements that indicate which page should be displayed next. Refer to Section 9.6, "Struts Configuration File" for an example of the `struts-config.xml` file.

Interface pages are constructed using the Tiles component of the Struts Framework. The layout file (`/WEB-INF/tiles-def.xml`) contains "definitions" for the various pages. Refer to Section 9.5, "Interface Page Configuration File" for an example of the `tiles.def.xml` file. In order to deploy Java Server Pages (JSP) files, you must add them to the OAAM shared library. See Chapter 7, "Customizing Oracle Adaptive Access Manager" for more information about the Oracle Adaptive Access Manager Extensions Shared Library.

9.2 Customizing the OAAM Interface Flow and JSP Layout

To customize the OAAM user interface flow and the layout of the Java Server Pages (JSPs), you must override the OAAM Server JSP and struts action targets using the OAAM Extensions Shared library. The Extensions Shared Library contains the following two files to be used for the customizations:

- `WEB-INF/struts-config-extension.xml`
- `WEB-INF/tiles-def-extension.xml`

> **Note:** Customizations should only be done in the OAAM Extensions Shared Library. Do not modify the `struts-config.xml` and `tiles-def.xml` files.

9.3 Customizing Java Server Pages (JSPs)

To customize the look and feel presented in the graphical user interface (GUI), add the custom JSP files to the OAAM Extensions shared library and then add the definitions to the `tiles-def-extension.xml` file.

The following example shows the definition for the password page, as defined in `tiles-defs.xml`:

```
<definition name="password" extends="bharosa.uio.baseLayout">
  <put name="body" value="/password.jsp"/>
</definition>
```

At run time the password page dynamically displays all necessary GUI elements for the user to enter the required credential.

The following example shows the definition of a custom password page that can be added to `tiles-def-extension.xml`:

```
<definition name="password" extends="bharosa.uio.baseLayout">
  <put name="body" value="/customPassword.jsp"/>
</definition>
```

If the definition is added to the `tiles-def-extension` file, the new `customPassword.jsp` is used when the OAAM Server attempts to display the "password" page.

9.4 Overriding Struts Definitions

Similar to overriding JSP content files, the struts action classes and their mappings could be overridden.

The following example shows the definition for the login action, as defined in `struts-config.xml`:

```
<action path="/login" type="com.bharosa.uio.actions.LoginAction">
     <forward name="success" path="/updateLoginStatus.do" redirect="true"/>
     <forward name="loginJump" path="/loginJumpPage.jsp" redirect="true"/>
     <forward name="password" path="password"/>
     <forward name="challenge" path="/challengeUser.do" redirect="true"/>
   </action>
```

The following example shows the possible values you could override for the login action by using `struts-config.xml`:

```
<action path="/login" type="com.bharosa.uio.actions.CustomLoginAction">
     <forward name="success" path="/updateLoginStatus.do" redirect="true"/>
     <forward name="loginJump" path="/customLoginJumpPage.jsp" redirect="true"/>
     <forward name="password" path="password"/>
     <forward name="challenge" path="/customChallengeUser.do" redirect="true"/>
   </action>
```

9.5 Interface Page Configuration File

The user interface pages are constructed using the Tiles component of the Struts Framework.

9.5.1 Rendering the Page

The following example extends the baseLayout definition and uses a JSP named `registerQuestionsHTML.jsp` to render the content tile. It renders content appropriate for the JSP named `registerQuestionsHTML.jsp`:

```
<definition name="registerQuestionsHTML" extends="bharosa.uio.baseLayout">
   <put name="body" value="/registerQuestionsHTML.jsp"/
  </definition>
```

The rendered page consists of content from the body tile. The body tile contains the output from `registerQuestionsHTML.jsp`.

9.5.2 tiles-def.xml

This section shows a `tiles-def.xml file`.

```
<tiles-definitions>

  <!-- ========================================================== -->
  <!-- Master definition - Start                                  -->
  <!-- ========================================================== -->
  <!-- Main page layout used as a root for other page definitions -->

  <definition name="bharosa.uio.baseLayout" path="/bharosaUIOBaseLayout.jsp">
    <put name="header" value="/header.jsp"/>
    <put name="footer" value="/footer.jsp"/>
    <put name="body" value="${body}"/>
  </definition>

  <definition name="bharosa.uio.messageLayout"
path="/bharosaUIOMessageLayout.jsp">
    <put name="body" value="${body}"/>
  </definition>

  <!-- login success -->

  <definition name="loginSuccess" extends="bharosa.uio.baseLayout">
    <put name="body" value="/loginSuccess.jsp"/>
  </definition>

  <!-- login fail -->
  <definition name="loginFail" extends="bharosa.uio.baseLayout">
    <put name="body" value="/loginFail.jsp"/>
  </definition>

  <!-- password entry -->
  <definition name="password" extends="bharosa.uio.baseLayout">
    <put name="body" value="/password.jsp"/>
  </definition>

  <!-- register questions -->
  <definition name="registerInfo" extends="bharosa.uio.baseLayout">
    <put name="body" value="/registerInfo.jsp"/>
  </definition>

  <definition name="registerAuthenticator" extends="bharosa.uio.baseLayout">
    <put name="body" value="/registerAuthenticator.jsp"/>
  </definition>

  <definition name="registerQuestions" extends="bharosa.uio.baseLayout">
    <put name="body" value="/registerQuestions.jsp"/>
  </definition>

  <definition name="registerQuestionsHTML" extends="bharosa.uio.baseLayout">
    <put name="body" value="/registerQuestionsHTML.jsp"/>
```

```
  </definition>

  <definition name="registerUserInfo" extends="bharosa.uio.baseLayout">
    <put name="body" value="/registerUserInfo.jsp"/>
  </definition>

  <definition name="userPreferences" extends="bharosa.uio.baseLayout">
    <put name="body" value="/userPreferences.jsp"/>
  </definition>

  <definition name="registrationRequired" extends="bharosa.uio.baseLayout">
    <put name="body" value="/registrationRequired.jsp"/>
  </definition>

  <definition name="changePassword" extends="bharosa.uio.baseLayout">
    <put name="body" value="/changePassword.jsp"/>
  </definition>

  <definition name="forgotPassword" extends="bharosa.uio.baseLayout">
    <put name="body" value="/forgotPassword.jsp"/>
  </definition>

  <definition name="userInput" extends="bharosa.uio.baseLayout">
    <put name="body" value="/userInput.jsp"/>
  </definition>

  <!-- challenge User -->
  <definition name="challengeUser" extends="bharosa.uio.baseLayout">
    <put name="body" value="/challengeUser.jsp"/>
  </definition>

  <definition name="challengeUserForgotPassword" extends="bharosa.uio.baseLayout">
    <put name="body" value="/challengeUser.jsp"/>
  </definition>

  <definition name="challengeWait" extends="bharosa.uio.baseLayout">
    <put name="body" value="/challengeWait.jsp"/>
  </definition>

  <!-- qaExists -->
  <definition name="qaExists" extends="bharosa.uio.baseLayout">
    <put name="body" value="/qaExists.jsp"/>
  </definition>

  <!-- register qa done -->
  <definition name="questRegistered" extends="bharosa.uio.baseLayout">
    <put name="body" value="/registerQAdone.jsp"/>
  </definition>

  <!-- signon -->
  <definition name="signon" extends="bharosa.uio.baseLayout">
    <put name="body" value="/securityProfile.jsp"/>
  </definition>

  <!-- messages -->
  <definition name="message" extends="bharosa.uio.messageLayout">
    <put name="body" value="/message.jsp"/>
  </definition>
</tiles-definitions>
```

9.6 Struts Configuration File

The Struts framework drives the navigation between the user interface pages.

9.6.1 Action Path

The action definition includes the path, which defines what the URL will be. The login page example is shown.

```
<action path="/login" type="com.bharosa.uio.actions.LoginAction">
     <forward name="success" path="/updateLoginStatus.do" redirect="true"/>
     <forward name="loginJump" path="/loginJumpPage.jsp" redirect="true"/>
     <forward name="password" path="password"/>
     <forward name="challenge" path="/challengeUser.do" redirect="true"/>
   </action>
```

9.6.2 Action Type

In login page example, the URL is `http://<server name>/oaam_server/login.do`. The `login.do` comes from the path definition of "`/login`."

The type parameter defines the class that performs the action.The following classes are provided with the sample user pages.

Table 9–1 Action Type Classes

Class Name	Description
com.bharosa.uio.actions.LoginAction	
com.bharosa.uio.actions.LoginFailAction	Displays error message in OAAM Server page. For example, the page could display a login blocked message.
com.bharosa.uio.actions.ActivityAction	
com.bharosa.uio.actions.PasswordAction	
com.bharosa.uio.actions.UpdateAuthStatusAction	Updates the user authentication status and, if appropriate, it triggers pattern data processing.
com.bharosa.uio.actions.ValidateTrxAction	
com.bharosa.uio.actions.FlashFingerprintAction	
com.bharosa.uio.actions.LogoutAction	Logs out the user session and redirects to login page
com.bharosa.uio.actions.SignOnAction	
com.bharosa.uio.actions.RegisterQuestionsAction	Displays sets of questions which the user can choose and register the correct answer for each.
com.bharosa.uio.actions.ChangePasswordAction	
com.bharosa.uio.actions.ForgotPasswordAction	
com.bharosa.uio.actions.UserInputAction	
com.bharosa.uio.actions.UserPreferencesDoneAction	
com.bharosa.uio.actions.ChallengeUserAction	Challenges the user by displaying a question-pad with one of the questions already registered by the user
com.bharosa.uio.actions.ChangeUserNameAction	

Table 9–1 (Cont.) Action Type Classes

Class Name	Description
com.bharosa.uio.actions.MessageAction	
com.bharosa.uio.actions.ExitAction	
com.bharosa.uio.actions.ErrorAction	

9.6.3 Struts Configuration File

This section shows a `struts-config.xml` file.

```
<struts-config>

  <!-- ========== Global Forward Definitions ============================== -->

  <global-forwards>
    <forward name="session_expired" path="/error.do?action=session_expired"
redirect="true"/>
    <forward name="emptyLoginId" path="/error.do?action=empty" redirect="true"/>
    <forward name="fail" path="/error.do?action=fail" redirect="true"/>
    <forward name="invalid_user" path="/error.do?action=invalid_user"
redirect="true"/>
    <forward name="error" path="/error.do?action=error" redirect="true"/>
    <forward name="block" path="/error.do?action=block" redirect="true"/>
    <forward name="challenge_block" path="/error.do?action=block"
redirect="true"/>
    <forward name="cookieDisabled" path="/error.do?action=cookieDisabled"
redirect="true"/>
    <forward name="accessDenied" path="/error.do?action=accessDenied"
redirect="true"/>
    <forward name="invalid_request" path="/error.do?action=accessDenied"
redirect="true"/>
    <forward name="user_disabled" path="/error.do?action=disabled"
redirect="true"/>
    <forward name="wrong_answer" path="/error.do?action=wrong_answer"
redirect="true"/>
    <forward name="login" path="/error.do" redirect="true"/>
  </global-forwards>

  <!-- ========== Action Mapping Definitions ============================== -->
  <action-mappings>

    <!-- action mappings for login -->

    <action path="/login" type="com.bharosa.uio.actions.LoginAction">
      <forward name="success" path="/updateLoginStatus.do" redirect="true"/>
      <forward name="loginJump" path="/loginJumpPage.jsp" redirect="true"/>
      <forward name="password" path="password"/>
      <forward name="passwordFT" path="password"/>
      <forward name="challenge" path="/challengeUser.do" redirect="true"/>
    </action>

    <action path="/loginFail" type="com.bharosa.uio.actions.LoginFailAction">
      <forward name="success" path="loginFail"/>
    </action>

    <action path="/activity" type="com.bharosa.uio.actions.ActivityAction">
      <forward name="success" path="loginSuccess" redirect="true"/>
    </action>
```

```
    <!-- validate password -->

    <action path="/password" type="com.bharosa.uio.actions.PasswordAction">
      <forward name="success" path="/exit.do"/>
      <forward name="invalid_user" path="/updateLoginStatus.do" />
      <forward name="noproxy" path="/updateLoginStatus.do"/>
      <forward name="resetPassword" path="/expiredPassword.do" redirect="true" />
    </action>

    <action path="/updateLoginStatus"
type="com.bharosa.uio.actions.UpdateAuthStatusAction">
      <forward name="success" path="/exit.do"/>
      <forward name="challenge" path="/challengeUser.do" redirect="true"/>
      <forward name="registerUser" path="/registerQuestions.do" redirect="true"/>
      <forward name="registerAuthenticator" path="/registerImage.do"
redirect="true"/>
      <forward name="registerQuestions" path="/registerQuestions.do"
redirect="true"/>
      <forward name="registerQuestionsHTML" path="/registerQuestions.do"
redirect="true"/>
      <forward name="registerUserInfo" path="/registerUserInfo.do"
redirect="true"/>
      <forward name="signon" path="signon" redirect="true"/>
    </action>

    <action path="/updateForgotPasswordStatus"
type="com.bharosa.uio.actions.UpdateAuthStatusAction" parameter="ForgotPassword">
      <forward name="success" path="/resetPassword.do" redirect="true" />
      <forward name="challenge" path="/challengeUserForgotPassword.do"
redirect="true"/>
      <forward name="registerUser" path="/registerQuestions.do" redirect="true"/>
      <forward name="registerQuestions" path="/registerQuestions.do"
redirect="true"/>
      <forward name="registerQuestionsHTML" path="/registerQuestions.do"
redirect="true"/>
      <forward name="registerUserInfo" path="/registerUserInfo.do"
redirect="true"/>
      <forward name="signon" path="signon" redirect="true"/>
    </action>

      <action path="/validateTrx"
type="com.bharosa.uio.actions.ValidateTrxAction">
          <forward name="success" path="/exit.do"/>
          <forward name="challenge" path="/challengeUserTrx.do" redirect="true"/>
      </action>

    <action path="/flashFingerprint"
type="com.bharosa.uio.actions.FlashFingerprintAction">
       <forward name="success" path="/flashFingerprint.jsp"/>
     </action>

      <!-- action mappings for logout -->

    <action path="/logout" type="com.bharosa.uio.actions.LogoutAction">
      <forward name="success" path="/loginPage.jsp" />
    </action>

    <!-- action mappings for signon -->
```

```
    <action path="/signon" type="com.bharosa.uio.actions.SignOnAction">
      <forward name="securityProfile" path="/securityProfile.jsp"
redirect="true"/>
      <forward name="securityDone" path="/activity.do" redirect="true"/>
    </action>

    <!-- action mappings for security QA -->

    <action path="/registerQuestions"
type="com.bharosa.uio.actions.RegisterQuestionsAction">
      <forward name="qaExists" path="qaExists" redirect="true"/>
      <forward name="registerAuthenticator" path="registerAuthenticator"/>
      <forward name="registerQuestions" path="registerQuestions"/>
      <forward name="registerQuestionsHTML" path="registerQuestionsHTML"/>
      <forward name="registerInfo" path="registerInfo"/>
      <forward name="registerUserInfo" path="registerUserInfo"/>
      <forward name="skip" path="/exit.do"/>
      <forward name="success" path="/exit.do"/>
    </action>

    <action path="/registerImage"
type="com.bharosa.uio.actions.RegisterQuestionsAction" parameter="RegisterImage">
      <forward name="registerAuthenticator" path="registerAuthenticator"/>
      <forward name="success" path="/exit.do"/>
    </action>

    <action path="/registerUserInfo"
type="com.bharosa.uio.actions.RegisterQuestionsAction"
parameter="RegisterUserInfo">
      <forward name="registerUserInfo" path="registerUserInfo"/>
      <forward name="success" path="/exit.do"/>
    </action>

    <action path="/userPreferences"
type="com.bharosa.uio.actions.RegisterQuestionsAction"
parameter="UserPreferences">
      <forward name="registerAuthenticator" path="userPreferences"/>
      <forward name="registerInfo" path="userPreferences"/>
      <forward name="registerQuestions" path="registerQuestions"/>
      <forward name="registerQuestionsHTML" path="registerQuestionsHTML"/>
      <forward name="registerUserInfo" path="registerUserInfo" />
      <forward name="changePassword" path="/changePassword.do" />
      <forward name="success" path="userPreferences"/>
      <forward name="registrationRequired" path="registrationRequired"/>
      <forward name="exit" path="/exit.do" />
    </action>

    <action path="/changePassword"
type="com.bharosa.uio.actions.ChangePasswordAction">
      <forward name="changePassword" path="changePassword" />
      <forward name="success" path="/userPreferences.do" redirect="true" />
      <forward name="exit" path="/exit.do" />
    </action>

    <action path="/resetPassword"
type="com.bharosa.uio.actions.ChangePasswordAction" parameter="ResetPassword">
      <forward name="changePassword" path="changePassword" />
      <forward name="success" path="/exit.do" />
      <forward name="updateStatus" path="/updateLoginStatus.do" redirect="true" />
    </action>
```

```
    <action path="/expiredPassword"
type="com.bharosa.uio.actions.ChangePasswordAction" parameter="ExpiredPassword">
      <forward name="changePassword" path="changePassword" />
      <forward name="success" path="/exit.do" />
      <forward name="updateStatus" path="/updateLoginStatus.do" redirect="true" />
    </action>

    <action path="/forgotPassword"
type="com.bharosa.uio.actions.ForgotPasswordAction">
      <forward name="forgotPassword" path="forgotPassword" />
      <forward name="challenge" path="/challengeUserForgotPassword.do" />
      <forward name="success" path="/exit.do" />
      <forward name="noproxy" path="/updateForgotPasswordStatus.do" />
    </action>

    <action path="/getUserInput" type="com.bharosa.uio.actions.UserInputAction">
      <forward name="showAuthenticator" path="userInput" />
      <forward name="success" path="/exit.do" />
    </action>


    <action path="/userPreferencesDone"
type="com.bharosa.uio.actions.UserPreferencesDoneAction">
      <forward name="success" path="/exit.do"/>
      <forward name="exit" path="/exit.do" />
    </action>
    <!-- action mappings for challenge user -->

    <action path="/challengeUser"
type="com.bharosa.uio.actions.ChallengeUserAction">
      <forward name="success" path="/exit.do" />
      <forward name="challenge" path="challengeUser"/>
      <forward name="registerUser" path="/registerQuestions.do" redirect="true"/>
      <forward name="registerAuthenticator" path="/registerImage.do"
redirect="true"/>
      <forward name="registerQuestions" path="/registerQuestions.do"
redirect="true"/>
      <forward name="registerQuestionsHTML" path="/registerQuestions.do"
redirect="true"/>
      <forward name="registerUserInfo" path="/registerUserInfo.do"
redirect="true"/>
      <forward name="wait" path="challengeWait"/>
    </action>

    <action path="/challengeUserTrx"
type="com.bharosa.uio.actions.ChallengeUserAction" parameter="transaction">
      <forward name="success" path="/exit.do" />
      <forward name="challenge" path="challengeUser"/>
      <forward name="registerUser" path="/registerQuestions.do" redirect="true"/>
      <forward name="registerAuthenticator" path="/registerImage.do"
redirect="true"/>
      <forward name="registerQuestions" path="/registerQuestions.do"
redirect="true"/>
      <forward name="registerQuestionsHTML" path="/registerQuestions.do"
redirect="true"/>
      <forward name="registerUserInfo" path="/registerUserInfo.do"
redirect="true"/>
      <forward name="wait" path="challengeWait"/>
    </action>
```

```
    <action path="/challengeUserForgotPassword"
type="com.bharosa.uio.actions.ChallengeUserAction" parameter="ForgotPassword">
      <forward name="success" path="/resetPassword.do" redirect="true"/>
      <forward name="forgotPassword" path="forgotPassword" />
      <forward name="challenge" path="challengeUserForgotPassword"/>
      <forward name="wait" path="challengeWait"/>
    </action>

    <action path="/changeUserId"
type="com.bharosa.uio.actions.ChangeUserNameAction">
      <forward name="success" path="/exit.do" />
    </action>

    <!-- action mappings for message -->

    <action path="/message" type="com.bharosa.uio.actions.MessageAction">
      <forward name="success" path="message"/>
    </action>

    <action path="/exit" type="com.bharosa.uio.actions.ExitAction">
      <forward name="success" path="/empty.jsp"/>
    </action>

    <action path="/error" type="com.bharosa.uio.actions.ErrorAction">
      <forward name="login" path="/loginPage.jsp" redirect="true" />
    </action>

  </action-mappings>

  <!--The Tiles Request Processor for processing all the Tile requests-->
  <controller processorClass="org.apache.struts.tiles.TilesRequestProcessor"/>

  <!-- message resources -->
  <message-resources parameter="proxyweb" null="false"/>

  <!-- tiles plug-in -->
  <plug-in className="org.apache.struts.tiles.TilesPlugin">
    <set-property property="definitions-config"
value="/WEB-INF/tiles-def.xml,/WEB-INF/tiles-def-extension.xml"/>
    <set-property property="definitions-debug" value="0"/>
    <set-property property="definitions-parser-details" value="0"/>
    <set-property property="definitions-parser-validate" value="true"/>
    <set-property property="moduleAware" value="true"/>
  </plug-in>

</struts-config>
```

10

Using Virtual Authentication Devices

Oracle Adaptive Access Manager includes unique functionality to protect end users while interacting with a protected web application. The virtual authentication devices are used to protect users during the process of entering and transmitting authentication credentials and provide them with verification they are authenticating on the valid application. Each virtual authentication device (VAD) has its own unique set of security features that make it much more than a mere image on a web page.

This chapter contains the following sections:

10.1 Terminology

This section defines terms used in this chapter.

Table 10–1 VAD Terminology

Term	Description
Authenticator / Authentipad	A control for user input included in OAAM that provides a keyboard and enables personalization.
Personalization	Assigning an image and generated phrase during registration. The phrase and image provide end users with verification they are authenticating on the valid application.
Virtual Keypad/Keyboard	A method for user input where the user clicks screen keys instead of an external keyboard.
Jitter	The act of moving key location slightly on each time the authenticator is generated.
Offset	The act of moving a whole key set on screen.
Key Randomization	The act of randomizing the key order.
Timestamp	A string generated from the current system time or client side time.
Masking	Replacing characters in an HTML input field.

10.2 Virtual Authentication Devices and Set of 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. 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.

10.3 Virtual Authentication Types

The following authentication devices are described in this section:

- TextPad
- PinPad
- QuestionPad
- Keypad

10.3.1 TextPad

TextPad is a personalized device for entering a password or PIN using a regular keyboard. This method of data entry helps to defend against phishing primarily. TextPad is often deployed as the default for all users in a large deployment. Then, each user individually can upgrade to another device if he wishes. The personal image and phrase a user registers and sees every time he logs in to the valid site serves as a shared secret between the user and server. If this shared secret is not presented or presented incorrectly, the users will notice. An example TextPad is shown in Figure 10–1.

Figure 10–1 TextPad

10.3.2 PinPad

PinPad is a lightweight authentication device for entering a numeric PIN. An example PinPad is shown in Figure 10–2.

Figure 10–2 PinPad

10.3.3 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 10–3.

Figure 10–3 QuestionPad

10.3.4 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 10–4.

Figure 10–4 KeyPad

10.4 Authenticator Composition

An authenticator is comprised of a number of elements. These elements are combined at runtime to produce the Authenticator for display on the client side.

Table 10–2 Elements of an authenticator

Element	Description
Personalized Image	An image selected by the user during registration. This is stored in the user repository in OAAM.
Authenticator Frame	An image that forms the frame of the authenticator. It contains graphics to represent user controls.
Timestamp, Phrase and Keyset	Image elements that are generated to build the personalization of the authenticator.
HTML Controls	A set of JavaScript controlled HTML elements for data entry and submission of data.

10.5 Virtual Authentication Device Properties

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

10.5.1 Property Files Used in the Authenticator's Configuration

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`).

10.5.2 TextPad Authenticator Properties

Table 10–3 lists the TextPad Authenticator Properties

Table 10–3 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

10.5.3 PinPad Authenticator Properties

Table 10–4 lists the PinPad Authenticator Properties

Table 10–4 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

10.5.4 QuestionPad Authenticator Properties

Table 10–5 lists the QuestionPad Authenticator Properties

Table 10–5 QuestionPad Authenticator Properties

Feature	Property
Default BG (Can be application specific)	bharosa.uio.<appId>.DeviceQuestionPad.default.image = textpad_bg/UIO_BG.jpg
Challenge Frame File (Can be application specific)	bharosa.uio.<appId>.<challengeType>.DeviceQuestionPad.frame = Note: Challenge type can be any configured challenge type (ChallengeQuestion, ChallengeEmail, and others)

10.5.5 KeyPad Authenticator Properties

Table 10–6 lists the KeyPad Authenticator Properties

Table 10–6 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

10.5.6 Frame Design and Element Positioning

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.

10.5.6.1 Background Images

For the background images to be displayed in the virtual authentication device, set the following property:

```
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.

10.5.6.2 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.

KeySets are defined by a series user defined enums.

The first enum defines the rows of the KeySet and points to 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 10–7 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 10–8 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.

10.5.6.3 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
```

10.5.6.4 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
```

10.5.6.5 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.

10.5.6.6 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 = ffffff
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.capsshiftimg=kp_v2_first_caps.jpg
```

10.5.7 Customization Steps

The process is as follows:

1. Add virtual authentication device related properties and custom KeySet related enum properties to `bharosa_server.properties` and save it in the `<temp-folder>/WEB-INF/classes` folder. Refer to the rest of the chapter for more information on defining keysets and other virtual authentication device properties.
2. Add key image files to `<temp-folder>/WEB-INF/classes/bharosa_properties/<pad>_skins`.
3. Add Frame Image Files: `<temp-folder>//WEB-INF/classes/bharosa_properties/<pad>_bg`.
4. Create OAAM Extensions Shared Library using `bharosa_server.properties`.
5. Deploy the custom OAAM Extensions Shared Library into both the OAAM Managed Servers (OAAM Admin and OAAM Server).
   a. Re-Jar the war using the command:

   ```
   jar -cvfm oracle.oaam.extensions.war <temp-folder>/META-INF/MANIFEST.MF -C
   <temp-folder>
   ```

   Note: Make sure original MANIFEST.MF remains same as that contains shared library information.

   b. Re-deploy the updated `oracle.oaam.extensions.war` as a shared library with targets as oaam_server and oaam_admin.
6. Restart OAAM Servers and validate your changes by accessing application.

10.6 Displaying Virtual Authentication Devices

This section describes the flow to render virtual authentication devices. It contains the following topics:

- Setting Up Before Calling the get<pad type> Method
- Getting the AuthentiPads
- Setting Properties After Getting Authentipad Object
- Displaying Virtual Authentication Devices

10.6.1 Setting Up Before Calling the get<pad type> Method

In order to get the `bgFile`, you need to obtain it from the user by performing:

```
String bgFile = (String)
authUser.getSecurityPreferences().get("imagePath");
```

10.6.2 Getting the AuthentiPads

The main API that handles authentipad generation is `BharosaClientImpl.getInstance().get<pad type>`.

The following methods can be used to get commonly used AuthentiPads:

- `BharosaClientImpl.getInstance().getFullKeyPad(...)`
- `BharosaClientImpl.getInstance().getAlphaNumericKeyPad(...)`
- `BharosaClientImpl.getInstance().getTextPad(...)`
- `BharosaClientImpl.getInstance().getQuestionPad(...)`
- `BharosaClientImpl.getInstance().getPinPad(...)`

Each method takes the same set of parameters:

Table 10–9 Authentipad: Method Parameters

Parameter	Description
String padName	Identifier of the AuthentiPad, used in the HTML as the base name of input fields and JavaScript variables.
String frameFile	Image path to use for the frame.
String backgroundFile	Image path to use for the background image. If using OAAM assignment APIs, OAAM stores the users assigned image in the VCryptAuthUser object: `(String) authUser.getSecurityPreferences().get("imagePath")`
VCryptLocalizedString captionText	A localized string to display as the caption on the AuthentiPad ■ `VCryptLocalizedString(String, VCryptLocale)` ■ `VCryptLocalizedString(String, Locale)` ■ `VCryptLocalizedString(String)`
boolean isADACompliant	Flag to designate if the AuthentiPad should be rendered with extra text and links for screen readers.
boolean hasJS	Flag to designate if the user has JavaScript enabled.
boolean hasImages	Flag to designate if the user has images enabled.

10.6.3 Setting Properties After Getting Authentipad Object

You need to set timestamp, timezone and display only property to the authentipad object that was obtained.

The following table shows fields that may need to be set on the AuthentiPad once it is created:

Table 10–10 Authentipad: Setting Additional Fields

Parameter	Description
authentiPad.setTimeStamp(Date timeStamp)	Sets the timestamp to display on the pad.
authentiPad.setTimeZone(TimeZone timeZone)	Sets the timezone to display on the pad.
authentiPad.setDisplayOnly(boolean displayOnly)	Flag to designate if the pad should be rendered without interactive fields and links. Commonly used to during image registration.
authentiPad.setQuestionText(VCryptLocalizedString questionText)	Used to display question on a QuestionPad.

10.6.4 Displaying Virtual Authentication Devices

VADs are rendered in an HTML page. Any page that is to render a VAD must include the `bharosa_pad.js` JavaScript file. The `bharosa_pad.js` file is a JavaScript library for rendering VADs and handling user interaction.

To get the HTML / JavaScript render string to be placed into an HTML page, call `authentiPad.getHTML()`.

The output of this method, will be an HTML string containing required image maps and JavaScript constructors required to display the VAD.

Once rendered, the VAD will make a request for the image to be displayed. The URL used to render the image is configured by the property: `bharosa.authentipad.image.url`.

10.7 Enabling Accessible Versions of Authenticators

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
```

10.8 Localizing Virtual Authentication Device in OAAM 11*g*

This section contains the following topics:

- Overview
- Example using German Locale

10.8.1 Overview

The process is as follows:

1. Create the `client_resource_<locale>.properties` file with virtual authentication device related properties and save it in the `<temp-folder>/WEB-INF/classes` folder.
2. Add the custom keyset related enum properties to `bharosa_server.properties` and save it in the `<temp-folder>/WEB-INF/classes` folder. Refer to the rest of the chapter for more information on defining keysets and other virtual authentication device properties.
3. Add key image files to `<temp-folder>/WEB-INF/classes/bharosa_properties/alphapad_skins_<locale>`.
4. Add Frame Image Files: `<temp-folder>//WEB-INF/classes/bharosa_properties/alphapad_bg`.

5. Create OAAM Extensions Shared Library using `client_resource_<locale>.properties` and `bharosa_server.properties`.
6. Deploy the custom OAAM Extensions Shared Library into both the OAAM Managed Servers (OAAM Admin and OAAM Server).
7. Test the localized keypads.

10.8.2 Example using German Locale

An example of localizing the pads in German is shown below:

1. Unzip the OAAM Extensions shared library war file into a temp directory `<temp-folder>`.
2. Create `client_resource_de.properties` in `<temp-folder>/WEB-INF/classes/` if not already present
3. Add these in `client_resource_de.properties`

```
# Keyset to use for German locale
bharosa.authentipad.keypad.default.keyset=german

 # Caption Coordinates for new German Pad
bharosa.authentipad.keypad.caption.y = 330
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

# Frame files to use for new German Pad
bharosa.authentipad.keypad.frame.file=alphapad_bg/kp_frame_03.png
bharosa.authentipad.keypad.sample.frame.file=alphapad_bg/kp_frame_03.png
bharosa.uio.default.register.DeviceKeyPadFull.frame = alphapad_bg/kp_frame_
03.png
bharosa.uio.default.userpreferences.DeviceKeyPadFull.frame = alphapad_bg/kp_
frame_03.png

# Skins directory containing German key images
bharosa.authentipad.keypad.skins.dirlist=alphapad_skins_de/square


# Timestamp Coordinates for new German Pad
bharosa.authentipad.keypad.timestamp.y = 330
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 = ffffff
bharosa.authentipad.keypad.timestamp.font.type= 0
bharosa.authentipad.keypad.timestamp.font.size = 9
```

4. Create bharosa_server.properties in <temp-folder>/WEB-INF/classes if not already present.

```
######################### German Full Keypad Keyset
############################
```

```
bharosa.authentipad.keypad.german.keyset.enum=German KeyPad Keyset Enum
bharosa.authentipad.keypad.german.keyset.enum.row1=0
bharosa.authentipad.keypad.german.keyset.enum.row1.name=German KeyPad Keyset
Row 1
bharosa.authentipad.keypad.german.keyset.enum.row1.description=German KeyPad
Keyset Row 1
bharosa.authentipad.keypad.german.keyset.enum.row1.keys=bharosa.authentipad.key
pad.german.keyset.row1.enum
bharosa.authentipad.keypad.german.keyset.enum.row1.order=1

bharosa.authentipad.keypad.german.keyset.enum.row2=1
bharosa.authentipad.keypad.german.keyset.enum.row2.name=German KeyPad Keyset
Row 2
bharosa.authentipad.keypad.german.keyset.enum.row2.description=German KeyPad
Keyset Row 2
bharosa.authentipad.keypad.german.keyset.enum.row2.keys=bharosa.authentipad.key
pad.german.keyset.row2.enum
bharosa.authentipad.keypad.german.keyset.enum.row2.order=2

bharosa.authentipad.keypad.german.keyset.enum.row3=2
bharosa.authentipad.keypad.german.keyset.enum.row3.name=German KeyPad Keyset
Row 3
bharosa.authentipad.keypad.german.keyset.enum.row3.description=German KeyPad
Keyset Row 3
bharosa.authentipad.keypad.german.keyset.enum.row3.keys=bharosa.authentipad.key
pad.german.keyset.row3.enum
bharosa.authentipad.keypad.german.keyset.enum.row3.order=3

bharosa.authentipad.keypad.german.keyset.enum.row4=3
bharosa.authentipad.keypad.german.keyset.enum.row4.name=German KeyPad Keyset
Row 4
bharosa.authentipad.keypad.german.keyset.enum.row4.description=German KeyPad
Keyset Row 4
bharosa.authentipad.keypad.german.keyset.enum.row4.keys=bharosa.authentipad.key
pad.german.keyset.row4.enum
bharosa.authentipad.keypad.german.keyset.enum.row4.order=4

bharosa.authentipad.keypad.german.keyset.enum.row5=4
bharosa.authentipad.keypad.german.keyset.enum.row5.name=German KeyPad Keyset
Row 5
bharosa.authentipad.keypad.german.keyset.enum.row5.description=German KeyPad
Keyset Row 5
bharosa.authentipad.keypad.german.keyset.enum.row5.keys=bharosa.authentipad.key
pad.german.keyset.row5.enum
bharosa.authentipad.keypad.german.keyset.enum.row5.order=5

##############################################\u00C0 to \u00FF Keyset
#######################################################

bharosa.authentipad.keypad.german.keyset.enum=German KeyPad Keyset Enum
bharosa.authentipad.keypad.german.keyset.enum.row6=5
bharosa.authentipad.keypad.german.keyset.enum.row6.name=German KeyPad Keyset
Row 6
bharosa.authentipad.keypad.german.keyset.enum.row6.description=German KeyPad
Keyset Row 6
bharosa.authentipad.keypad.german.keyset.enum.row6.keys=bharosa.authentipad.key
pad.german.keyset.row6.enum
bharosa.authentipad.keypad.german.keyset.enum.row6.order=6
```

```
bharosa.authentipad.keypad.german.keyset.enum.row7=6
bharosa.authentipad.keypad.german.keyset.enum.row7.name=German KeyPad Keyset
Row 7
bharosa.authentipad.keypad.german.keyset.enum.row7.description=German KeyPad
Keyset Row 7
bharosa.authentipad.keypad.german.keyset.enum.row7.keys=bharosa.authentipad.key
pad.german.keyset.row7.enum
bharosa.authentipad.keypad.german.keyset.enum.row7.order=7

bharosa.authentipad.keypad.german.keyset.enum.row8=7
bharosa.authentipad.keypad.german.keyset.enum.row8.name=German KeyPad Keyset
Row 8
bharosa.authentipad.keypad.german.keyset.enum.row8.description=German KeyPad
Keyset Row 8
bharosa.authentipad.keypad.german.keyset.enum.row8.keys=bharosa.authentipad.key
pad.german.keyset.row8.enum
bharosa.authentipad.keypad.german.keyset.enum.row8.order=8

bharosa.authentipad.keypad.german.keyset.enum.row9=8
bharosa.authentipad.keypad.german.keyset.enum.row9.name=German KeyPad Keyset
Row 9
bharosa.authentipad.keypad.german.keyset.enum.row9.description=German KeyPad
Keyset Row 9
bharosa.authentipad.keypad.german.keyset.enum.row9.keys=bharosa.authentipad.key
pad.german.keyset.row9.enum
bharosa.authentipad.keypad.german.keyset.enum.row9.order=9

bharosa.authentipad.keypad.german.keyset.enum.row10=9
bharosa.authentipad.keypad.german.keyset.enum.row10.name=German KeyPad Keyset
Row 10
bharosa.authentipad.keypad.german.keyset.enum.row10.description=German KeyPad
Keyset Row 10
bharosa.authentipad.keypad.german.keyset.enum.row10.keys=bharosa.authentipad.ke
ypad.german.keyset.row10.enum
bharosa.authentipad.keypad.german.keyset.enum.row10.order=10


################################################################################
##########################



bharosa.authentipad.keypad.german.keyset.row1.enum=German KeyPad Keyset Row 1
bharosa.authentipad.keypad.german.keyset.row1.enum.key1=0
bharosa.authentipad.keypad.german.keyset.row1.enum.key1.name=!
bharosa.authentipad.keypad.german.keyset.row1.enum.key1.description=!
bharosa.authentipad.keypad.german.keyset.row1.enum.key1.value=!
bharosa.authentipad.keypad.german.keyset.row1.enum.key1.shiftvalue=!
bharosa.authentipad.keypad.german.keyset.row1.enum.key1.image=kp_v2_exclaim.png
bharosa.authentipad.keypad.german.keyset.row1.enum.key1.order=1

bharosa.authentipad.keypad.german.keyset.row1.enum.key2=1
bharosa.authentipad.keypad.german.keyset.row1.enum.key2.name=@
bharosa.authentipad.keypad.german.keyset.row1.enum.key2.description=@
bharosa.authentipad.keypad.german.keyset.row1.enum.key2.value=@
bharosa.authentipad.keypad.german.keyset.row1.enum.key2.shiftvalue=@
bharosa.authentipad.keypad.german.keyset.row1.enum.key2.image=kp_v2_rate.png
bharosa.authentipad.keypad.german.keyset.row1.enum.key2.order=2

bharosa.authentipad.keypad.german.keyset.row1.enum.key3=2
```

```
bharosa.authentipad.keypad.german.keyset.row1.enum.key3.name=#
bharosa.authentipad.keypad.german.keyset.row1.enum.key3.description=#
bharosa.authentipad.keypad.german.keyset.row1.enum.key3.value=#
bharosa.authentipad.keypad.german.keyset.row1.enum.key3.shiftvalue=#
bharosa.authentipad.keypad.german.keyset.row1.enum.key3.image=kp_v2_hash.png
bharosa.authentipad.keypad.german.keyset.row1.enum.key3.order=3

bharosa.authentipad.keypad.german.keyset.row1.enum.key4=3
bharosa.authentipad.keypad.german.keyset.row1.enum.key4.name=$
bharosa.authentipad.keypad.german.keyset.row1.enum.key4.description=$
bharosa.authentipad.keypad.german.keyset.row1.enum.key4.value=$
bharosa.authentipad.keypad.german.keyset.row1.enum.key4.shiftvalue=$
bharosa.authentipad.keypad.german.keyset.row1.enum.key4.image=kp_v2_dollar.png
bharosa.authentipad.keypad.german.keyset.row1.enum.key4.order=4

bharosa.authentipad.keypad.german.keyset.row1.enum.key5=4
bharosa.authentipad.keypad.german.keyset.row1.enum.key5.name=%
bharosa.authentipad.keypad.german.keyset.row1.enum.key5.description=%
bharosa.authentipad.keypad.german.keyset.row1.enum.key5.value=%
bharosa.authentipad.keypad.german.keyset.row1.enum.key5.shiftvalue=%
bharosa.authentipad.keypad.german.keyset.row1.enum.key5.image=kp_v2_percent.png
bharosa.authentipad.keypad.german.keyset.row1.enum.key5.order=5

bharosa.authentipad.keypad.german.keyset.row1.enum.key6=5
bharosa.authentipad.keypad.german.keyset.row1.enum.key6.name=^
bharosa.authentipad.keypad.german.keyset.row1.enum.key6.description=^
bharosa.authentipad.keypad.german.keyset.row1.enum.key6.value=^
bharosa.authentipad.keypad.german.keyset.row1.enum.key6.shiftvalue=^
bharosa.authentipad.keypad.german.keyset.row1.enum.key6.image=kp_v2_carat.png
bharosa.authentipad.keypad.german.keyset.row1.enum.key6.order=6

bharosa.authentipad.keypad.german.keyset.row1.enum.key7=6
bharosa.authentipad.keypad.german.keyset.row1.enum.key7.name=&
bharosa.authentipad.keypad.german.keyset.row1.enum.key7.description=&
bharosa.authentipad.keypad.german.keyset.row1.enum.key7.value=&
bharosa.authentipad.keypad.german.keyset.row1.enum.key7.shiftvalue=&
bharosa.authentipad.keypad.german.keyset.row1.enum.key7.image=kp_v2_and.png
bharosa.authentipad.keypad.german.keyset.row1.enum.key7.order=7

bharosa.authentipad.keypad.german.keyset.row1.enum.key8=7
bharosa.authentipad.keypad.german.keyset.row1.enum.key8.name=*
bharosa.authentipad.keypad.german.keyset.row1.enum.key8.description=*
bharosa.authentipad.keypad.german.keyset.row1.enum.key8.value=*
bharosa.authentipad.keypad.german.keyset.row1.enum.key8.shiftvalue=*
bharosa.authentipad.keypad.german.keyset.row1.enum.key8.image=kp_v2_
asterisk.png
bharosa.authentipad.keypad.german.keyset.row1.enum.key8.order=8

bharosa.authentipad.keypad.german.keyset.row1.enum.key9=8
bharosa.authentipad.keypad.german.keyset.row1.enum.key9.name=(
bharosa.authentipad.keypad.german.keyset.row1.enum.key9.description=(
bharosa.authentipad.keypad.german.keyset.row1.enum.key9.value=(
bharosa.authentipad.keypad.german.keyset.row1.enum.key9.shiftvalue=(
bharosa.authentipad.keypad.german.keyset.row1.enum.key9.image=kp_v2_
leftbraces.png
bharosa.authentipad.keypad.german.keyset.row1.enum.key9.order=9

bharosa.authentipad.keypad.german.keyset.row1.enum.key10=9
bharosa.authentipad.keypad.german.keyset.row1.enum.key10.name=)
bharosa.authentipad.keypad.german.keyset.row1.enum.key10.description=)
```

```
bharosa.authentipad.keypad.german.keyset.row1.enum.key10.value=)
bharosa.authentipad.keypad.german.keyset.row1.enum.key10.shiftvalue=)
bharosa.authentipad.keypad.german.keyset.row1.enum.key10.image=kp_v2_
rightbraces.png
bharosa.authentipad.keypad.german.keyset.row1.enum.key10.order=10

bharosa.authentipad.keypad.german.keyset.row1.enum.key11=10
bharosa.authentipad.keypad.german.keyset.row1.enum.key11.name=_
bharosa.authentipad.keypad.german.keyset.row1.enum.key11.description=_
bharosa.authentipad.keypad.german.keyset.row1.enum.key11.value=_
bharosa.authentipad.keypad.german.keyset.row1.enum.key11.shiftvalue=_
bharosa.authentipad.keypad.german.keyset.row1.enum.key11.image=kp_v2_
underscore.png
bharosa.authentipad.keypad.german.keyset.row1.enum.key11.order=11

bharosa.authentipad.keypad.german.keyset.row1.enum.key12=11
bharosa.authentipad.keypad.german.keyset.row1.enum.key12.name=+
bharosa.authentipad.keypad.german.keyset.row1.enum.key12.description=+
bharosa.authentipad.keypad.german.keyset.row1.enum.key12.value=+
bharosa.authentipad.keypad.german.keyset.row1.enum.key12.shiftvalue=+
bharosa.authentipad.keypad.german.keyset.row1.enum.key12.image=kp_v2_plus.png
bharosa.authentipad.keypad.german.keyset.row1.enum.key12.order=12

bharosa.authentipad.keypad.german.keyset.row1.enum.key13=12
bharosa.authentipad.keypad.german.keyset.row1.enum.key13.name=~
bharosa.authentipad.keypad.german.keyset.row1.enum.key13.description=~
bharosa.authentipad.keypad.german.keyset.row1.enum.key13.value=~
bharosa.authentipad.keypad.german.keyset.row1.enum.key13.shiftvalue=~
bharosa.authentipad.keypad.german.keyset.row1.enum.key13.image=kp_v2_tilda.png
bharosa.authentipad.keypad.german.keyset.row1.enum.key13.order=13


bharosa.authentipad.keypad.german.keyset.row2.enum=German KeyPad Keyset Row 2
bharosa.authentipad.keypad.german.keyset.row2.enum.key1=0
bharosa.authentipad.keypad.german.keyset.row2.enum.key1.name=1
bharosa.authentipad.keypad.german.keyset.row2.enum.key1.description=1
bharosa.authentipad.keypad.german.keyset.row2.enum.key1.value=1
bharosa.authentipad.keypad.german.keyset.row2.enum.key1.shiftvalue=1
bharosa.authentipad.keypad.german.keyset.row2.enum.key1.image=kp_v2_1.png
bharosa.authentipad.keypad.german.keyset.row2.enum.key1.order=1

bharosa.authentipad.keypad.german.keyset.row2.enum.key2=1
bharosa.authentipad.keypad.german.keyset.row2.enum.key2.name=2
bharosa.authentipad.keypad.german.keyset.row2.enum.key2.description=2
bharosa.authentipad.keypad.german.keyset.row2.enum.key2.value=2
bharosa.authentipad.keypad.german.keyset.row2.enum.key2.shiftvalue=2
bharosa.authentipad.keypad.german.keyset.row2.enum.key2.image=kp_v2_2.png
bharosa.authentipad.keypad.german.keyset.row2.enum.key2.order=2

bharosa.authentipad.keypad.german.keyset.row2.enum.key3=2
bharosa.authentipad.keypad.german.keyset.row2.enum.key3.name=3
bharosa.authentipad.keypad.german.keyset.row2.enum.key3.description=3
bharosa.authentipad.keypad.german.keyset.row2.enum.key3.value=3
bharosa.authentipad.keypad.german.keyset.row2.enum.key3.shiftvalue=3
bharosa.authentipad.keypad.german.keyset.row2.enum.key3.image=kp_v2_3.png
bharosa.authentipad.keypad.german.keyset.row2.enum.key3.order=3

bharosa.authentipad.keypad.german.keyset.row2.enum.key4=3
bharosa.authentipad.keypad.german.keyset.row2.enum.key4.name=4
bharosa.authentipad.keypad.german.keyset.row2.enum.key4.description=4
```

```
bharosa.authentipad.keypad.german.keyset.row2.enum.key4.value=4
bharosa.authentipad.keypad.german.keyset.row2.enum.key4.shiftvalue=4
bharosa.authentipad.keypad.german.keyset.row2.enum.key4.image=kp_v2_4.png
bharosa.authentipad.keypad.german.keyset.row2.enum.key4.order=4

bharosa.authentipad.keypad.german.keyset.row2.enum.key5=4
bharosa.authentipad.keypad.german.keyset.row2.enum.key5.name=5
bharosa.authentipad.keypad.german.keyset.row2.enum.key5.description=5
bharosa.authentipad.keypad.german.keyset.row2.enum.key5.value=5
bharosa.authentipad.keypad.german.keyset.row2.enum.key5.shiftvalue=5
bharosa.authentipad.keypad.german.keyset.row2.enum.key5.image=kp_v2_5.png
bharosa.authentipad.keypad.german.keyset.row2.enum.key5.order=5

bharosa.authentipad.keypad.german.keyset.row2.enum.key6=5
bharosa.authentipad.keypad.german.keyset.row2.enum.key6.name=6
bharosa.authentipad.keypad.german.keyset.row2.enum.key6.description=6
bharosa.authentipad.keypad.german.keyset.row2.enum.key6.value=6
bharosa.authentipad.keypad.german.keyset.row2.enum.key6.shiftvalue=6
bharosa.authentipad.keypad.german.keyset.row2.enum.key6.image=kp_v2_6.png
bharosa.authentipad.keypad.german.keyset.row2.enum.key6.order=6

bharosa.authentipad.keypad.german.keyset.row2.enum.key7=6
bharosa.authentipad.keypad.german.keyset.row2.enum.key7.name=7
bharosa.authentipad.keypad.german.keyset.row2.enum.key7.description=7
bharosa.authentipad.keypad.german.keyset.row2.enum.key7.value=7
bharosa.authentipad.keypad.german.keyset.row2.enum.key7.shiftvalue=7
bharosa.authentipad.keypad.german.keyset.row2.enum.key7.image=kp_v2_7.png
bharosa.authentipad.keypad.german.keyset.row2.enum.key7.order=7

bharosa.authentipad.keypad.german.keyset.row2.enum.key8=7
bharosa.authentipad.keypad.german.keyset.row2.enum.key8.name=8
bharosa.authentipad.keypad.german.keyset.row2.enum.key8.description=8
bharosa.authentipad.keypad.german.keyset.row2.enum.key8.value=8
bharosa.authentipad.keypad.german.keyset.row2.enum.key8.shiftvalue=8
bharosa.authentipad.keypad.german.keyset.row2.enum.key8.image=kp_v2_8.png
bharosa.authentipad.keypad.german.keyset.row2.enum.key8.order=8

bharosa.authentipad.keypad.german.keyset.row2.enum.key9=8
bharosa.authentipad.keypad.german.keyset.row2.enum.key9.name=9
bharosa.authentipad.keypad.german.keyset.row2.enum.key9.description=9
bharosa.authentipad.keypad.german.keyset.row2.enum.key9.value=9
bharosa.authentipad.keypad.german.keyset.row2.enum.key9.shiftvalue=9
bharosa.authentipad.keypad.german.keyset.row2.enum.key9.image=kp_v2_9.png
bharosa.authentipad.keypad.german.keyset.row2.enum.key9.order=9

bharosa.authentipad.keypad.german.keyset.row2.enum.key10=9
bharosa.authentipad.keypad.german.keyset.row2.enum.key10.name=0
bharosa.authentipad.keypad.german.keyset.row2.enum.key10.description=0
bharosa.authentipad.keypad.german.keyset.row2.enum.key10.value=0
bharosa.authentipad.keypad.german.keyset.row2.enum.key10.shiftvalue=0
bharosa.authentipad.keypad.german.keyset.row2.enum.key10.image=kp_v2_0.png
bharosa.authentipad.keypad.german.keyset.row2.enum.key10.order=10

bharosa.authentipad.keypad.german.keyset.row2.enum.key11=10
bharosa.authentipad.keypad.german.keyset.row2.enum.key11.name=-
bharosa.authentipad.keypad.german.keyset.row2.enum.key11.description=-
bharosa.authentipad.keypad.german.keyset.row2.enum.key11.value=-
bharosa.authentipad.keypad.german.keyset.row2.enum.key11.shiftvalue=-
bharosa.authentipad.keypad.german.keyset.row2.enum.key11.image=kp_v2_hyphen.png
bharosa.authentipad.keypad.german.keyset.row2.enum.key11.order=11
```

```
bharosa.authentipad.keypad.german.keyset.row2.enum.key12=11
bharosa.authentipad.keypad.german.keyset.row2.enum.key12.name==
bharosa.authentipad.keypad.german.keyset.row2.enum.key12.description==
bharosa.authentipad.keypad.german.keyset.row2.enum.key12.value==
bharosa.authentipad.keypad.german.keyset.row2.enum.key12.shiftvalue==
bharosa.authentipad.keypad.german.keyset.row2.enum.key12.image=kp_v2_equals.png
bharosa.authentipad.keypad.german.keyset.row2.enum.key12.order=12

bharosa.authentipad.keypad.german.keyset.row2.enum.key13=12
bharosa.authentipad.keypad.german.keyset.row2.enum.key13.name=`
bharosa.authentipad.keypad.german.keyset.row2.enum.key13.description=`
bharosa.authentipad.keypad.german.keyset.row2.enum.key13.value=`
bharosa.authentipad.keypad.german.keyset.row2.enum.key13.shiftvalue=`
bharosa.authentipad.keypad.german.keyset.row2.enum.key13.image=kp_v2_apost.png
bharosa.authentipad.keypad.german.keyset.row2.enum.key13.order=13

bharosa.authentipad.keypad.german.keyset.row3.enum=German KeyPad Keyset Row 3
bharosa.authentipad.keypad.german.keyset.row3.enum.key1=0
bharosa.authentipad.keypad.german.keyset.row3.enum.key1.name=q
bharosa.authentipad.keypad.german.keyset.row3.enum.key1.description=q
bharosa.authentipad.keypad.german.keyset.row3.enum.key1.value=q
bharosa.authentipad.keypad.german.keyset.row3.enum.key1.shiftvalue=Q
bharosa.authentipad.keypad.german.keyset.row3.enum.key1.image=kp_v2_Q.png
bharosa.authentipad.keypad.german.keyset.row3.enum.key1.order=1

bharosa.authentipad.keypad.german.keyset.row3.enum.key2=1
bharosa.authentipad.keypad.german.keyset.row3.enum.key2.name=w
bharosa.authentipad.keypad.german.keyset.row3.enum.key2.description=w
bharosa.authentipad.keypad.german.keyset.row3.enum.key2.value=w
bharosa.authentipad.keypad.german.keyset.row3.enum.key2.shiftvalue=W
bharosa.authentipad.keypad.german.keyset.row3.enum.key2.image=kp_v2_W.png
bharosa.authentipad.keypad.german.keyset.row3.enum.key2.order=2

bharosa.authentipad.keypad.german.keyset.row3.enum.key3=2
bharosa.authentipad.keypad.german.keyset.row3.enum.key3.name=e
bharosa.authentipad.keypad.german.keyset.row3.enum.key3.description=e
bharosa.authentipad.keypad.german.keyset.row3.enum.key3.value=e
bharosa.authentipad.keypad.german.keyset.row3.enum.key3.shiftvalue=E
bharosa.authentipad.keypad.german.keyset.row3.enum.key3.image=kp_v2_E.png
bharosa.authentipad.keypad.german.keyset.row3.enum.key3.order=3

bharosa.authentipad.keypad.german.keyset.row3.enum.key4=3
bharosa.authentipad.keypad.german.keyset.row3.enum.key4.name=r
bharosa.authentipad.keypad.german.keyset.row3.enum.key4.description=r
bharosa.authentipad.keypad.german.keyset.row3.enum.key4.value=r
bharosa.authentipad.keypad.german.keyset.row3.enum.key4.shiftvalue=R
bharosa.authentipad.keypad.german.keyset.row3.enum.key4.image=kp_v2_R.png
bharosa.authentipad.keypad.german.keyset.row3.enum.key4.order=4

bharosa.authentipad.keypad.german.keyset.row3.enum.key5=4
bharosa.authentipad.keypad.german.keyset.row3.enum.key5.name=t
bharosa.authentipad.keypad.german.keyset.row3.enum.key5.description=t
bharosa.authentipad.keypad.german.keyset.row3.enum.key5.value=t
bharosa.authentipad.keypad.german.keyset.row3.enum.key5.shiftvalue=T
bharosa.authentipad.keypad.german.keyset.row3.enum.key5.image=kp_v2_T.png
bharosa.authentipad.keypad.german.keyset.row3.enum.key5.order=5

bharosa.authentipad.keypad.german.keyset.row3.enum.key6=5
bharosa.authentipad.keypad.german.keyset.row3.enum.key6.name=y
```

```
bharosa.authentipad.keypad.german.keyset.row3.enum.key6.description=y
bharosa.authentipad.keypad.german.keyset.row3.enum.key6.value=y
bharosa.authentipad.keypad.german.keyset.row3.enum.key6.shiftvalue=Y
bharosa.authentipad.keypad.german.keyset.row3.enum.key6.image=kp_v2_Y.png
bharosa.authentipad.keypad.german.keyset.row3.enum.key6.order=6

bharosa.authentipad.keypad.german.keyset.row3.enum.key7=6
bharosa.authentipad.keypad.german.keyset.row3.enum.key7.name=u
bharosa.authentipad.keypad.german.keyset.row3.enum.key7.description=u
bharosa.authentipad.keypad.german.keyset.row3.enum.key7.value=u
bharosa.authentipad.keypad.german.keyset.row3.enum.key7.shiftvalue=U
bharosa.authentipad.keypad.german.keyset.row3.enum.key7.image=kp_v2_U.png
bharosa.authentipad.keypad.german.keyset.row3.enum.key7.order=7

bharosa.authentipad.keypad.german.keyset.row3.enum.key8=7
bharosa.authentipad.keypad.german.keyset.row3.enum.key8.name=i
bharosa.authentipad.keypad.german.keyset.row3.enum.key8.description=i
bharosa.authentipad.keypad.german.keyset.row3.enum.key8.value=i
bharosa.authentipad.keypad.german.keyset.row3.enum.key8.shiftvalue=I
bharosa.authentipad.keypad.german.keyset.row3.enum.key8.image=kp_v2_I.png
bharosa.authentipad.keypad.german.keyset.row3.enum.key8.order=8

bharosa.authentipad.keypad.german.keyset.row3.enum.key9=8
bharosa.authentipad.keypad.german.keyset.row3.enum.key9.name=o
bharosa.authentipad.keypad.german.keyset.row3.enum.key9.description=o
bharosa.authentipad.keypad.german.keyset.row3.enum.key9.value=o
bharosa.authentipad.keypad.german.keyset.row3.enum.key9.shiftvalue=O
bharosa.authentipad.keypad.german.keyset.row3.enum.key9.image=kp_v2_O.png
bharosa.authentipad.keypad.german.keyset.row3.enum.key9.order=9

bharosa.authentipad.keypad.german.keyset.row3.enum.key10=9
bharosa.authentipad.keypad.german.keyset.row3.enum.key10.name=p
bharosa.authentipad.keypad.german.keyset.row3.enum.key10.description=p
bharosa.authentipad.keypad.german.keyset.row3.enum.key10.value=p
bharosa.authentipad.keypad.german.keyset.row3.enum.key10.shiftvalue=P
bharosa.authentipad.keypad.german.keyset.row3.enum.key10.image=kp_v2_P.png
bharosa.authentipad.keypad.german.keyset.row3.enum.key10.order=10

bharosa.authentipad.keypad.german.keyset.row3.enum.key11=10
bharosa.authentipad.keypad.german.keyset.row3.enum.key11.name={
bharosa.authentipad.keypad.german.keyset.row3.enum.key11.description={
bharosa.authentipad.keypad.german.keyset.row3.enum.key11.value={
bharosa.authentipad.keypad.german.keyset.row3.enum.key11.shiftvalue={
bharosa.authentipad.keypad.german.keyset.row3.enum.key11.image=kp_v2_
leftcurlybraces.png
bharosa.authentipad.keypad.german.keyset.row3.enum.key11.order=11

bharosa.authentipad.keypad.german.keyset.row3.enum.key12=11
bharosa.authentipad.keypad.german.keyset.row3.enum.key12.name=}
bharosa.authentipad.keypad.german.keyset.row3.enum.key12.description=}
bharosa.authentipad.keypad.german.keyset.row3.enum.key12.value=}
bharosa.authentipad.keypad.german.keyset.row3.enum.key12.shiftvalue=}
bharosa.authentipad.keypad.german.keyset.row3.enum.key12.image=kp_v2_
rightcurlybraces.png
bharosa.authentipad.keypad.german.keyset.row3.enum.key12.order=12

bharosa.authentipad.keypad.german.keyset.row3.enum.key13=12
bharosa.authentipad.keypad.german.keyset.row3.enum.key13.name="
bharosa.authentipad.keypad.german.keyset.row3.enum.key13.description="
bharosa.authentipad.keypad.german.keyset.row3.enum.key13.value="
```

```
bharosa.authentipad.keypad.german.keyset.row3.enum.key13.shiftvalue="
bharosa.authentipad.keypad.german.keyset.row3.enum.key13.image=kp_v2_quotes.png
bharosa.authentipad.keypad.german.keyset.row3.enum.key13.order=13


bharosa.authentipad.keypad.german.keyset.row4.enum=German KeyPad Keyset Row 4
bharosa.authentipad.keypad.german.keyset.row4.enum.key1=0
bharosa.authentipad.keypad.german.keyset.row4.enum.key1.name=a
bharosa.authentipad.keypad.german.keyset.row4.enum.key1.description=a
bharosa.authentipad.keypad.german.keyset.row4.enum.key1.value=a
bharosa.authentipad.keypad.german.keyset.row4.enum.key1.shiftvalue=A
bharosa.authentipad.keypad.german.keyset.row4.enum.key1.image=kp_v2_A.png
bharosa.authentipad.keypad.german.keyset.row4.enum.key1.order=1

bharosa.authentipad.keypad.german.keyset.row4.enum.key2=1
bharosa.authentipad.keypad.german.keyset.row4.enum.key2.name=s
bharosa.authentipad.keypad.german.keyset.row4.enum.key2.description=s
bharosa.authentipad.keypad.german.keyset.row4.enum.key2.value=s
bharosa.authentipad.keypad.german.keyset.row4.enum.key2.shiftvalue=S
bharosa.authentipad.keypad.german.keyset.row4.enum.key2.image=kp_v2_S.png
bharosa.authentipad.keypad.german.keyset.row4.enum.key2.order=2

bharosa.authentipad.keypad.german.keyset.row4.enum.key3=2
bharosa.authentipad.keypad.german.keyset.row4.enum.key3.name=d
bharosa.authentipad.keypad.german.keyset.row4.enum.key3.description=d
bharosa.authentipad.keypad.german.keyset.row4.enum.key3.value=d
bharosa.authentipad.keypad.german.keyset.row4.enum.key3.shiftvalue=D
bharosa.authentipad.keypad.german.keyset.row4.enum.key3.image=kp_v2_D.png
bharosa.authentipad.keypad.german.keyset.row4.enum.key3.order=3

bharosa.authentipad.keypad.german.keyset.row4.enum.key4=3
bharosa.authentipad.keypad.german.keyset.row4.enum.key4.name=f
bharosa.authentipad.keypad.german.keyset.row4.enum.key4.description=f
bharosa.authentipad.keypad.german.keyset.row4.enum.key4.value=f
bharosa.authentipad.keypad.german.keyset.row4.enum.key4.shiftvalue=F
bharosa.authentipad.keypad.german.keyset.row4.enum.key4.image=kp_v2_F.png
bharosa.authentipad.keypad.german.keyset.row4.enum.key4.order=4

bharosa.authentipad.keypad.german.keyset.row4.enum.key5=4
bharosa.authentipad.keypad.german.keyset.row4.enum.key5.name=g
bharosa.authentipad.keypad.german.keyset.row4.enum.key5.description=g
bharosa.authentipad.keypad.german.keyset.row4.enum.key5.value=g
bharosa.authentipad.keypad.german.keyset.row4.enum.key5.shiftvalue=G
bharosa.authentipad.keypad.german.keyset.row4.enum.key5.image=kp_v2_G.png
bharosa.authentipad.keypad.german.keyset.row4.enum.key5.order=5

bharosa.authentipad.keypad.german.keyset.row4.enum.key6=5
bharosa.authentipad.keypad.german.keyset.row4.enum.key6.name=h
bharosa.authentipad.keypad.german.keyset.row4.enum.key6.description=h
bharosa.authentipad.keypad.german.keyset.row4.enum.key6.value=h
bharosa.authentipad.keypad.german.keyset.row4.enum.key6.shiftvalue=H
bharosa.authentipad.keypad.german.keyset.row4.enum.key6.image=kp_v2_H.png
bharosa.authentipad.keypad.german.keyset.row4.enum.key6.order=6

bharosa.authentipad.keypad.german.keyset.row4.enum.key7=6
bharosa.authentipad.keypad.german.keyset.row4.enum.key7.name=j
bharosa.authentipad.keypad.german.keyset.row4.enum.key7.description=j
bharosa.authentipad.keypad.german.keyset.row4.enum.key7.value=j
bharosa.authentipad.keypad.german.keyset.row4.enum.key7.shiftvalue=J
bharosa.authentipad.keypad.german.keyset.row4.enum.key7.image=kp_v2_J.png
```

```
bharosa.authentipad.keypad.german.keyset.row4.enum.key7.order=7

bharosa.authentipad.keypad.german.keyset.row4.enum.key8=7
bharosa.authentipad.keypad.german.keyset.row4.enum.key8.name=k
bharosa.authentipad.keypad.german.keyset.row4.enum.key8.description=k
bharosa.authentipad.keypad.german.keyset.row4.enum.key8.value=k
bharosa.authentipad.keypad.german.keyset.row4.enum.key8.shiftvalue=K
bharosa.authentipad.keypad.german.keyset.row4.enum.key8.image=kp_v2_K.png
bharosa.authentipad.keypad.german.keyset.row4.enum.key8.order=8

bharosa.authentipad.keypad.german.keyset.row4.enum.key9=8
bharosa.authentipad.keypad.german.keyset.row4.enum.key9.name=l
bharosa.authentipad.keypad.german.keyset.row4.enum.key9.description=l
bharosa.authentipad.keypad.german.keyset.row4.enum.key9.value=l
bharosa.authentipad.keypad.german.keyset.row4.enum.key9.shiftvalue=L
bharosa.authentipad.keypad.german.keyset.row4.enum.key9.image=kp_v2_L.png
bharosa.authentipad.keypad.german.keyset.row4.enum.key9.order=9

bharosa.authentipad.keypad.german.keyset.row4.enum.key10=9
bharosa.authentipad.keypad.german.keyset.row4.enum.key10.name=:
bharosa.authentipad.keypad.german.keyset.row4.enum.key10.description=:
bharosa.authentipad.keypad.german.keyset.row4.enum.key10.value=:
bharosa.authentipad.keypad.german.keyset.row4.enum.key10.shiftvalue=:
bharosa.authentipad.keypad.german.keyset.row4.enum.key10.image=kp_v2_colon.png
bharosa.authentipad.keypad.german.keyset.row4.enum.key10.order=10

bharosa.authentipad.keypad.german.keyset.row4.enum.key11=10
bharosa.authentipad.keypad.german.keyset.row4.enum.key11.name=;
bharosa.authentipad.keypad.german.keyset.row4.enum.key11.description=;
bharosa.authentipad.keypad.german.keyset.row4.enum.key11.value=;
bharosa.authentipad.keypad.german.keyset.row4.enum.key11.shiftvalue=;
bharosa.authentipad.keypad.german.keyset.row4.enum.key11.image=kp_v2_
semicolon.png
bharosa.authentipad.keypad.german.keyset.row4.enum.key11.order=11

bharosa.authentipad.keypad.german.keyset.row4.enum.key12=11
bharosa.authentipad.keypad.german.keyset.row4.enum.key12.name=\\
bharosa.authentipad.keypad.german.keyset.row4.enum.key12.description=\\
bharosa.authentipad.keypad.german.keyset.row4.enum.key12.value=\\
bharosa.authentipad.keypad.german.keyset.row4.enum.key12.shiftvalue=\\
bharosa.authentipad.keypad.german.keyset.row4.enum.key12.image=kp_v2_
backslash.png
bharosa.authentipad.keypad.german.keyset.row4.enum.key12.order=12

bharosa.authentipad.keypad.german.keyset.row4.enum.key13=12
bharosa.authentipad.keypad.german.keyset.row4.enum.key13.name='
bharosa.authentipad.keypad.german.keyset.row4.enum.key13.description='
bharosa.authentipad.keypad.german.keyset.row4.enum.key13.value='
bharosa.authentipad.keypad.german.keyset.row4.enum.key13.shiftvalue='
bharosa.authentipad.keypad.german.keyset.row4.enum.key13.image=kp_v2_quote.png
bharosa.authentipad.keypad.german.keyset.row4.enum.key13.order=13

bharosa.authentipad.keypad.german.keyset.row5.enum=German KeyPad Keyset Row 5
bharosa.authentipad.keypad.german.keyset.row5.enum.key1=0
bharosa.authentipad.keypad.german.keyset.row5.enum.key1.name=z
bharosa.authentipad.keypad.german.keyset.row5.enum.key1.description=z
bharosa.authentipad.keypad.german.keyset.row5.enum.key1.value=z
bharosa.authentipad.keypad.german.keyset.row5.enum.key1.shiftvalue=Z
bharosa.authentipad.keypad.german.keyset.row5.enum.key1.image=kp_v2_Z.png
bharosa.authentipad.keypad.german.keyset.row5.enum.key1.order=1
```

```
bharosa.authentipad.keypad.german.keyset.row5.enum.key2=1
bharosa.authentipad.keypad.german.keyset.row5.enum.key2.name=x
bharosa.authentipad.keypad.german.keyset.row5.enum.key2.description=x
bharosa.authentipad.keypad.german.keyset.row5.enum.key2.value=x
bharosa.authentipad.keypad.german.keyset.row5.enum.key2.shiftvalue=X
bharosa.authentipad.keypad.german.keyset.row5.enum.key2.image=kp_v2_X.png
bharosa.authentipad.keypad.german.keyset.row5.enum.key2.order=2

bharosa.authentipad.keypad.german.keyset.row5.enum.key3=2
bharosa.authentipad.keypad.german.keyset.row5.enum.key3.name=c
bharosa.authentipad.keypad.german.keyset.row5.enum.key3.description=c
bharosa.authentipad.keypad.german.keyset.row5.enum.key3.value=c
bharosa.authentipad.keypad.german.keyset.row5.enum.key3.shiftvalue=C
bharosa.authentipad.keypad.german.keyset.row5.enum.key3.image=kp_v2_C.png
bharosa.authentipad.keypad.german.keyset.row5.enum.key3.order=3

bharosa.authentipad.keypad.german.keyset.row5.enum.key4=3
bharosa.authentipad.keypad.german.keyset.row5.enum.key4.name=v
bharosa.authentipad.keypad.german.keyset.row5.enum.key4.description=v
bharosa.authentipad.keypad.german.keyset.row5.enum.key4.value=v
bharosa.authentipad.keypad.german.keyset.row5.enum.key4.shiftvalue=V
bharosa.authentipad.keypad.german.keyset.row5.enum.key4.image=kp_v2_V.png
bharosa.authentipad.keypad.german.keyset.row5.enum.key4.order=4

bharosa.authentipad.keypad.german.keyset.row5.enum.key5=4
bharosa.authentipad.keypad.german.keyset.row5.enum.key5.name=b
bharosa.authentipad.keypad.german.keyset.row5.enum.key5.description=b
bharosa.authentipad.keypad.german.keyset.row5.enum.key5.value=b
bharosa.authentipad.keypad.german.keyset.row5.enum.key5.shiftvalue=B
bharosa.authentipad.keypad.german.keyset.row5.enum.key5.image=kp_v2_B.png
bharosa.authentipad.keypad.german.keyset.row5.enum.key5.order=5

bharosa.authentipad.keypad.german.keyset.row5.enum.key6=5
bharosa.authentipad.keypad.german.keyset.row5.enum.key6.name=n
bharosa.authentipad.keypad.german.keyset.row5.enum.key6.description=n
bharosa.authentipad.keypad.german.keyset.row5.enum.key6.value=n
bharosa.authentipad.keypad.german.keyset.row5.enum.key6.shiftvalue=N
bharosa.authentipad.keypad.german.keyset.row5.enum.key6.image=kp_v2_N.png
bharosa.authentipad.keypad.german.keyset.row5.enum.key6.order=6

bharosa.authentipad.keypad.german.keyset.row5.enum.key7=6
bharosa.authentipad.keypad.german.keyset.row5.enum.key7.name=m
bharosa.authentipad.keypad.german.keyset.row5.enum.key7.description=m
bharosa.authentipad.keypad.german.keyset.row5.enum.key7.value=m
bharosa.authentipad.keypad.german.keyset.row5.enum.key7.shiftvalue=M
bharosa.authentipad.keypad.german.keyset.row5.enum.key7.image=kp_v2_M.png
bharosa.authentipad.keypad.german.keyset.row5.enum.key7.order=7

bharosa.authentipad.keypad.german.keyset.row5.enum.key8=7
bharosa.authentipad.keypad.german.keyset.row5.enum.key8.name=<
bharosa.authentipad.keypad.german.keyset.row5.enum.key8.description=<
bharosa.authentipad.keypad.german.keyset.row5.enum.key8.value=<
bharosa.authentipad.keypad.german.keyset.row5.enum.key8.shiftvalue=<
bharosa.authentipad.keypad.german.keyset.row5.enum.key8.image=kp_v2_
lessthan.png
bharosa.authentipad.keypad.german.keyset.row5.enum.key8.order=8

bharosa.authentipad.keypad.german.keyset.row5.enum.key9=8
bharosa.authentipad.keypad.german.keyset.row5.enum.key9.name=>
```

```
bharosa.authentipad.keypad.german.keyset.row5.enum.key9.description=>
bharosa.authentipad.keypad.german.keyset.row5.enum.key9.value=>
bharosa.authentipad.keypad.german.keyset.row5.enum.key9.shiftvalue=>
bharosa.authentipad.keypad.german.keyset.row5.enum.key9.image=kp_v2_
greaterthan.png
bharosa.authentipad.keypad.german.keyset.row5.enum.key9.order=9

bharosa.authentipad.keypad.german.keyset.row5.enum.key10=9
bharosa.authentipad.keypad.german.keyset.row5.enum.key10.name=,
bharosa.authentipad.keypad.german.keyset.row5.enum.key10.description=,
bharosa.authentipad.keypad.german.keyset.row5.enum.key10.value=,
bharosa.authentipad.keypad.german.keyset.row5.enum.key10.shiftvalue=,
bharosa.authentipad.keypad.german.keyset.row5.enum.key10.image=kp_v2_comma.png
bharosa.authentipad.keypad.german.keyset.row5.enum.key10.order=10

bharosa.authentipad.keypad.german.keyset.row5.enum.key11=10
bharosa.authentipad.keypad.german.keyset.row5.enum.key11.name=.
bharosa.authentipad.keypad.german.keyset.row5.enum.key11.description=.
bharosa.authentipad.keypad.german.keyset.row5.enum.key11.value=.
bharosa.authentipad.keypad.german.keyset.row5.enum.key11.shiftvalue=.
bharosa.authentipad.keypad.german.keyset.row5.enum.key11.image=kp_v2_period.png
bharosa.authentipad.keypad.german.keyset.row5.enum.key11.order=11

bharosa.authentipad.keypad.german.keyset.row5.enum.key12=11
bharosa.authentipad.keypad.german.keyset.row5.enum.key12.name=/
bharosa.authentipad.keypad.german.keyset.row5.enum.key12.description=/
bharosa.authentipad.keypad.german.keyset.row5.enum.key12.value=/
bharosa.authentipad.keypad.german.keyset.row5.enum.key12.shiftvalue=/
bharosa.authentipad.keypad.german.keyset.row5.enum.key12.image=kp_v2_
forwardslash.png
bharosa.authentipad.keypad.german.keyset.row5.enum.key12.order=12

bharosa.authentipad.keypad.german.keyset.row5.enum.key13=12
bharosa.authentipad.keypad.german.keyset.row5.enum.key13.name=?
bharosa.authentipad.keypad.german.keyset.row5.enum.key13.description=?
bharosa.authentipad.keypad.german.keyset.row5.enum.key13.value=?
bharosa.authentipad.keypad.german.keyset.row5.enum.key13.shiftvalue=?
bharosa.authentipad.keypad.german.keyset.row5.enum.key13.image=kp_v2_
questionmark.png
bharosa.authentipad.keypad.german.keyset.row5.enum.key13.order=13




######################### Alternate Keypad Keyset
#############################################################

bharosa.authentipad.keypad.german.keyset.row6.enum=German KeyPad Keyset Row 6
bharosa.authentipad.keypad.german.keyset.row6.enum.key1=0
bharosa.authentipad.keypad.german.keyset.row6.enum.key1.name=\u00C0
bharosa.authentipad.keypad.german.keyset.row6.enum.key1.description=\u00C0
bharosa.authentipad.keypad.german.keyset.row6.enum.key1.value=\u00C0
bharosa.authentipad.keypad.german.keyset.row6.enum.key1.shiftvalue=\u00C0
bharosa.authentipad.keypad.german.keyset.row6.enum.key1.image=kp_v01_00C0.png
bharosa.authentipad.keypad.german.keyset.row6.enum.key1.order=1

bharosa.authentipad.keypad.german.keyset.row6.enum.key2=1
bharosa.authentipad.keypad.german.keyset.row6.enum.key2.name=\u00C1
```

```
bharosa.authentipad.keypad.german.keyset.row6.enum.key2.description=\u00C1
bharosa.authentipad.keypad.german.keyset.row6.enum.key2.value=\u00C1
bharosa.authentipad.keypad.german.keyset.row6.enum.key2.shiftvalue=\u00C1
bharosa.authentipad.keypad.german.keyset.row6.enum.key2.image=kp_v01_00C1.png
bharosa.authentipad.keypad.german.keyset.row6.enum.key2.order=2

bharosa.authentipad.keypad.german.keyset.row6.enum.key3=2
bharosa.authentipad.keypad.german.keyset.row6.enum.key3.name=\u00C2
bharosa.authentipad.keypad.german.keyset.row6.enum.key3.description=\u00C2
bharosa.authentipad.keypad.german.keyset.row6.enum.key3.value=\u00C2
bharosa.authentipad.keypad.german.keyset.row6.enum.key3.shiftvalue=\u00C2
bharosa.authentipad.keypad.german.keyset.row6.enum.key3.image=kp_v01_00C2.png
bharosa.authentipad.keypad.german.keyset.row6.enum.key3.order=3

bharosa.authentipad.keypad.german.keyset.row6.enum.key4=3
bharosa.authentipad.keypad.german.keyset.row6.enum.key4.name=\u00C3
bharosa.authentipad.keypad.german.keyset.row6.enum.key4.description=\u00C3
bharosa.authentipad.keypad.german.keyset.row6.enum.key4.value=\u00C3
bharosa.authentipad.keypad.german.keyset.row6.enum.key4.shiftvalue=\u00C3
bharosa.authentipad.keypad.german.keyset.row6.enum.key4.image=kp_v01_00C3.png
bharosa.authentipad.keypad.german.keyset.row6.enum.key4.order=4

bharosa.authentipad.keypad.german.keyset.row6.enum.key5=4
bharosa.authentipad.keypad.german.keyset.row6.enum.key5.name=\u00C4
bharosa.authentipad.keypad.german.keyset.row6.enum.key5.description=\u00C4
bharosa.authentipad.keypad.german.keyset.row6.enum.key5.value=\u00C4
bharosa.authentipad.keypad.german.keyset.row6.enum.key5.shiftvalue=\u00C4
bharosa.authentipad.keypad.german.keyset.row6.enum.key5.image=kp_v01_00C4.png
bharosa.authentipad.keypad.german.keyset.row6.enum.key5.order=5

bharosa.authentipad.keypad.german.keyset.row6.enum.key6=5
bharosa.authentipad.keypad.german.keyset.row6.enum.key6.name=\u00C5
bharosa.authentipad.keypad.german.keyset.row6.enum.key6.description=\u00C5
bharosa.authentipad.keypad.german.keyset.row6.enum.key6.value=\u00C5
bharosa.authentipad.keypad.german.keyset.row6.enum.key6.shiftvalue=\u00C5
bharosa.authentipad.keypad.german.keyset.row6.enum.key6.image=kp_v01_00C5.png
bharosa.authentipad.keypad.german.keyset.row6.enum.key6.order=6

bharosa.authentipad.keypad.german.keyset.row6.enum.key7=6
bharosa.authentipad.keypad.german.keyset.row6.enum.key7.name=\u00C6
bharosa.authentipad.keypad.german.keyset.row6.enum.key7.description=\u00C6
bharosa.authentipad.keypad.german.keyset.row6.enum.key7.value=\u00C6
bharosa.authentipad.keypad.german.keyset.row6.enum.key7.shiftvalue=\u00C6
bharosa.authentipad.keypad.german.keyset.row6.enum.key7.image=kp_v01_00C6.png
bharosa.authentipad.keypad.german.keyset.row6.enum.key7.order=7

bharosa.authentipad.keypad.german.keyset.row6.enum.key8=7
bharosa.authentipad.keypad.german.keyset.row6.enum.key8.name=\u00C7
bharosa.authentipad.keypad.german.keyset.row6.enum.key8.description=\u00C7
bharosa.authentipad.keypad.german.keyset.row6.enum.key8.value=\u00C7
bharosa.authentipad.keypad.german.keyset.row6.enum.key8.shiftvalue=\u00C7
bharosa.authentipad.keypad.german.keyset.row6.enum.key8.image=kp_v01_00C7.png
bharosa.authentipad.keypad.german.keyset.row6.enum.key8.order=8

bharosa.authentipad.keypad.german.keyset.row6.enum.key9=8
bharosa.authentipad.keypad.german.keyset.row6.enum.key9.name=\u00C8
bharosa.authentipad.keypad.german.keyset.row6.enum.key9.description=\u00C8
bharosa.authentipad.keypad.german.keyset.row6.enum.key9.value=\u00C8
bharosa.authentipad.keypad.german.keyset.row6.enum.key9.shiftvalue=\u00C8
bharosa.authentipad.keypad.german.keyset.row6.enum.key9.image=kp_v01_00C8.png
```

```
bharosa.authentipad.keypad.german.keyset.row6.enum.key9.order=9

bharosa.authentipad.keypad.german.keyset.row6.enum.key10=9
bharosa.authentipad.keypad.german.keyset.row6.enum.key10.name=\u00C9
bharosa.authentipad.keypad.german.keyset.row6.enum.key10.description=\u00C9
bharosa.authentipad.keypad.german.keyset.row6.enum.key10.value=\u00C9
bharosa.authentipad.keypad.german.keyset.row6.enum.key10.shiftvalue=\u00C9
bharosa.authentipad.keypad.german.keyset.row6.enum.key10.image=kp_v01_00C9.png
bharosa.authentipad.keypad.german.keyset.row6.enum.key10.order=10

bharosa.authentipad.keypad.german.keyset.row6.enum.key11=10
bharosa.authentipad.keypad.german.keyset.row6.enum.key11.name=\u00CA
bharosa.authentipad.keypad.german.keyset.row6.enum.key11.description=\u00CA
bharosa.authentipad.keypad.german.keyset.row6.enum.key11.value=\u00CA
bharosa.authentipad.keypad.german.keyset.row6.enum.key11.shiftvalue=\u00CA
bharosa.authentipad.keypad.german.keyset.row6.enum.key11.image=kp_v01_00CA.png
bharosa.authentipad.keypad.german.keyset.row6.enum.key11.order=11

bharosa.authentipad.keypad.german.keyset.row6.enum.key12=11
bharosa.authentipad.keypad.german.keyset.row6.enum.key12.name=\u00CB
bharosa.authentipad.keypad.german.keyset.row6.enum.key12.description=\u00CB
bharosa.authentipad.keypad.german.keyset.row6.enum.key12.value=\u00CB
bharosa.authentipad.keypad.german.keyset.row6.enum.key12.shiftvalue=\u00CB
bharosa.authentipad.keypad.german.keyset.row6.enum.key12.image=kp_v01_00CB.png
bharosa.authentipad.keypad.german.keyset.row6.enum.key12.order=12

bharosa.authentipad.keypad.german.keyset.row6.enum.key13=12
bharosa.authentipad.keypad.german.keyset.row6.enum.key13.name=\u00CC
bharosa.authentipad.keypad.german.keyset.row6.enum.key13.description=\u00CC
bharosa.authentipad.keypad.german.keyset.row6.enum.key13.value=\u00CC
bharosa.authentipad.keypad.german.keyset.row6.enum.key13.shiftvalue=\u00CC
bharosa.authentipad.keypad.german.keyset.row6.enum.key13.image=kp_v01_00CC.png
bharosa.authentipad.keypad.german.keyset.row6.enum.key13.order=13



bharosa.authentipad.keypad.german.keyset.row7.enum=German KeyPad Keyset Row 7
bharosa.authentipad.keypad.german.keyset.row7.enum.key1=0
bharosa.authentipad.keypad.german.keyset.row7.enum.key1.name=\u00CD
bharosa.authentipad.keypad.german.keyset.row7.enum.key1.description=\u00CD
bharosa.authentipad.keypad.german.keyset.row7.enum.key1.value=\u00CD
bharosa.authentipad.keypad.german.keyset.row7.enum.key1.shiftvalue=\u00CD
bharosa.authentipad.keypad.german.keyset.row7.enum.key1.image=kp_v01_00CD.png
bharosa.authentipad.keypad.german.keyset.row7.enum.key1.order=1

bharosa.authentipad.keypad.german.keyset.row7.enum.key2=1
bharosa.authentipad.keypad.german.keyset.row7.enum.key2.name=\u00CE
bharosa.authentipad.keypad.german.keyset.row7.enum.key2.description=\u00CE
bharosa.authentipad.keypad.german.keyset.row7.enum.key2.value=\u00CE
bharosa.authentipad.keypad.german.keyset.row7.enum.key2.shiftvalue=\u00CE
bharosa.authentipad.keypad.german.keyset.row7.enum.key2.image=kp_v01_00CE.png
bharosa.authentipad.keypad.german.keyset.row7.enum.key2.order=2

bharosa.authentipad.keypad.german.keyset.row7.enum.key3=2
bharosa.authentipad.keypad.german.keyset.row7.enum.key3.name=\u00CF
bharosa.authentipad.keypad.german.keyset.row7.enum.key3.description=\u00CF
bharosa.authentipad.keypad.german.keyset.row7.enum.key3.value=\u00CF
bharosa.authentipad.keypad.german.keyset.row7.enum.key3.shiftvalue=\u00CF
bharosa.authentipad.keypad.german.keyset.row7.enum.key3.image=kp_v01_00CF.png
```

```
bharosa.authentipad.keypad.german.keyset.row7.enum.key3.order=3

bharosa.authentipad.keypad.german.keyset.row7.enum.key4=3
bharosa.authentipad.keypad.german.keyset.row7.enum.key4.name=\u00D0
bharosa.authentipad.keypad.german.keyset.row7.enum.key4.description=\u00D0
bharosa.authentipad.keypad.german.keyset.row7.enum.key4.value=\u00D0
bharosa.authentipad.keypad.german.keyset.row7.enum.key4.shiftvalue=\u00D0
bharosa.authentipad.keypad.german.keyset.row7.enum.key4.image=kp_v01_00D0.png
bharosa.authentipad.keypad.german.keyset.row7.enum.key4.order=4

bharosa.authentipad.keypad.german.keyset.row7.enum.key5=4
bharosa.authentipad.keypad.german.keyset.row7.enum.key5.name=\u00D1
bharosa.authentipad.keypad.german.keyset.row7.enum.key5.description=\u00D1
bharosa.authentipad.keypad.german.keyset.row7.enum.key5.value=\u00D1
bharosa.authentipad.keypad.german.keyset.row7.enum.key5.shiftvalue=\u00D1
bharosa.authentipad.keypad.german.keyset.row7.enum.key5.image=kp_v01_00D1.png
bharosa.authentipad.keypad.german.keyset.row7.enum.key5.order=5

bharosa.authentipad.keypad.german.keyset.row7.enum.key6=5
bharosa.authentipad.keypad.german.keyset.row7.enum.key6.name=\u00D2
bharosa.authentipad.keypad.german.keyset.row7.enum.key6.description=\u00D2
bharosa.authentipad.keypad.german.keyset.row7.enum.key6.value=\u00D2
bharosa.authentipad.keypad.german.keyset.row7.enum.key6.shiftvalue=\u00D2
bharosa.authentipad.keypad.german.keyset.row7.enum.key6.image=kp_v01_00D2.png
bharosa.authentipad.keypad.german.keyset.row7.enum.key6.order=6

bharosa.authentipad.keypad.german.keyset.row7.enum.key7=6
bharosa.authentipad.keypad.german.keyset.row7.enum.key7.name=\u00D3
bharosa.authentipad.keypad.german.keyset.row7.enum.key7.description=\u00D3
bharosa.authentipad.keypad.german.keyset.row7.enum.key7.value=\u00D3
bharosa.authentipad.keypad.german.keyset.row7.enum.key7.shiftvalue=\u00D3
bharosa.authentipad.keypad.german.keyset.row7.enum.key7.image=kp_v01_00D3.png
bharosa.authentipad.keypad.german.keyset.row7.enum.key7.order=7

bharosa.authentipad.keypad.german.keyset.row7.enum.key8=7
bharosa.authentipad.keypad.german.keyset.row7.enum.key8.name=\u00D4
bharosa.authentipad.keypad.german.keyset.row7.enum.key8.description=\u00D4
bharosa.authentipad.keypad.german.keyset.row7.enum.key8.value=\u00D4
bharosa.authentipad.keypad.german.keyset.row7.enum.key8.shiftvalue=\u00D4
bharosa.authentipad.keypad.german.keyset.row7.enum.key8.image=kp_v01_00D4.png
bharosa.authentipad.keypad.german.keyset.row7.enum.key8.order=8

bharosa.authentipad.keypad.german.keyset.row7.enum.key9=8
bharosa.authentipad.keypad.german.keyset.row7.enum.key9.name=\u00D5
bharosa.authentipad.keypad.german.keyset.row7.enum.key9.description=\u00D5
bharosa.authentipad.keypad.german.keyset.row7.enum.key9.value=\u00D5
bharosa.authentipad.keypad.german.keyset.row7.enum.key9.shiftvalue=\u00D5
bharosa.authentipad.keypad.german.keyset.row7.enum.key9.image=kp_v01_00D5.png
bharosa.authentipad.keypad.german.keyset.row7.enum.key9.order=9

bharosa.authentipad.keypad.german.keyset.row7.enum.key10=9
bharosa.authentipad.keypad.german.keyset.row7.enum.key10.name=\u00D6
bharosa.authentipad.keypad.german.keyset.row7.enum.key10.description=\u00D6
bharosa.authentipad.keypad.german.keyset.row7.enum.key10.value=\u00D6
bharosa.authentipad.keypad.german.keyset.row7.enum.key10.shiftvalue=\u00D6
bharosa.authentipad.keypad.german.keyset.row7.enum.key10.image=kp_v01_00D6.png
bharosa.authentipad.keypad.german.keyset.row7.enum.key10.order=10

bharosa.authentipad.keypad.german.keyset.row7.enum.key11=10
bharosa.authentipad.keypad.german.keyset.row7.enum.key11.name=\u00D7
```

```
bharosa.authentipad.keypad.german.keyset.row7.enum.key11.description=\u00D7
bharosa.authentipad.keypad.german.keyset.row7.enum.key11.value=\u00D7
bharosa.authentipad.keypad.german.keyset.row7.enum.key11.shiftvalue=\u00D7
bharosa.authentipad.keypad.german.keyset.row7.enum.key11.image=kp_v01_00D7.png
bharosa.authentipad.keypad.german.keyset.row7.enum.key11.order=11

bharosa.authentipad.keypad.german.keyset.row7.enum.key12=11
bharosa.authentipad.keypad.german.keyset.row7.enum.key12.name=\u00D8
bharosa.authentipad.keypad.german.keyset.row7.enum.key12.description=\u00D8
bharosa.authentipad.keypad.german.keyset.row7.enum.key12.value=\u00D8
bharosa.authentipad.keypad.german.keyset.row7.enum.key12.shiftvalue=\u00D8
bharosa.authentipad.keypad.german.keyset.row7.enum.key12.image=kp_v01_00D8.png
bharosa.authentipad.keypad.german.keyset.row7.enum.key12.order=12

bharosa.authentipad.keypad.german.keyset.row7.enum.key13=12
bharosa.authentipad.keypad.german.keyset.row7.enum.key13.name=\u00D9
bharosa.authentipad.keypad.german.keyset.row7.enum.key13.description=\u00D9
bharosa.authentipad.keypad.german.keyset.row7.enum.key13.value=\u00D9
bharosa.authentipad.keypad.german.keyset.row7.enum.key13.shiftvalue=\u00D9
bharosa.authentipad.keypad.german.keyset.row7.enum.key13.image=kp_v01_00D9.png
bharosa.authentipad.keypad.german.keyset.row7.enum.key13.order=13



bharosa.authentipad.keypad.german.keyset.row8.enum=German KeyPad Keyset Row8
bharosa.authentipad.keypad.german.keyset.row8.enum.key1=0
bharosa.authentipad.keypad.german.keyset.row8.enum.key1.name=\u00DA
bharosa.authentipad.keypad.german.keyset.row8.enum.key1.description=\u00DA
bharosa.authentipad.keypad.german.keyset.row8.enum.key1.value=\u00DA
bharosa.authentipad.keypad.german.keyset.row8.enum.key1.shiftvalue=\u00DA
bharosa.authentipad.keypad.german.keyset.row8.enum.key1.image=kp_v01_00DA.png
bharosa.authentipad.keypad.german.keyset.row8.enum.key1.order=1

bharosa.authentipad.keypad.german.keyset.row8.enum.key2=1
bharosa.authentipad.keypad.german.keyset.row8.enum.key2.name=\u00DB
bharosa.authentipad.keypad.german.keyset.row8.enum.key2.description=\u00DB
bharosa.authentipad.keypad.german.keyset.row8.enum.key2.value=\u00DB
bharosa.authentipad.keypad.german.keyset.row8.enum.key2.shiftvalue=\u00DB
bharosa.authentipad.keypad.german.keyset.row8.enum.key2.image=kp_v01_00DB.png
bharosa.authentipad.keypad.german.keyset.row8.enum.key2.order=2

bharosa.authentipad.keypad.german.keyset.row8.enum.key3=2
bharosa.authentipad.keypad.german.keyset.row8.enum.key3.name=\u00DC
bharosa.authentipad.keypad.german.keyset.row8.enum.key3.description=\u00DC
bharosa.authentipad.keypad.german.keyset.row8.enum.key3.value=\u00DC
bharosa.authentipad.keypad.german.keyset.row8.enum.key3.shiftvalue=\u00DC
bharosa.authentipad.keypad.german.keyset.row8.enum.key3.image=kp_v01_00DC.png
bharosa.authentipad.keypad.german.keyset.row8.enum.key3.order=3

bharosa.authentipad.keypad.german.keyset.row8.enum.key4=3
bharosa.authentipad.keypad.german.keyset.row8.enum.key4.name=\u00DD
bharosa.authentipad.keypad.german.keyset.row8.enum.key4.description=\u00DD
bharosa.authentipad.keypad.german.keyset.row8.enum.key4.value=\u00DD
bharosa.authentipad.keypad.german.keyset.row8.enum.key4.shiftvalue=\u00DD
bharosa.authentipad.keypad.german.keyset.row8.enum.key4.image=kp_v01_00DD.png
bharosa.authentipad.keypad.german.keyset.row8.enum.key4.order=4

bharosa.authentipad.keypad.german.keyset.row8.enum.key5=4
bharosa.authentipad.keypad.german.keyset.row8.enum.key5.name=\u00DE
```

```
bharosa.authentipad.keypad.german.keyset.row8.enum.key5.description=\u00DE
bharosa.authentipad.keypad.german.keyset.row8.enum.key5.value=\u00DE
bharosa.authentipad.keypad.german.keyset.row8.enum.key5.shiftvalue=\u00DE
bharosa.authentipad.keypad.german.keyset.row8.enum.key5.image=kp_v01_00DE.png
bharosa.authentipad.keypad.german.keyset.row8.enum.key5.order=5

bharosa.authentipad.keypad.german.keyset.row8.enum.key6=5
bharosa.authentipad.keypad.german.keyset.row8.enum.key6.name=\u00DF
bharosa.authentipad.keypad.german.keyset.row8.enum.key6.description=\u00DF
bharosa.authentipad.keypad.german.keyset.row8.enum.key6.value=\u00DF
bharosa.authentipad.keypad.german.keyset.row8.enum.key6.shiftvalue=\u00DF
bharosa.authentipad.keypad.german.keyset.row8.enum.key6.image=kp_v01_00DF.png
bharosa.authentipad.keypad.german.keyset.row8.enum.key6.order=6

bharosa.authentipad.keypad.german.keyset.row8.enum.key7=6
bharosa.authentipad.keypad.german.keyset.row8.enum.key7.name=\u00E0
bharosa.authentipad.keypad.german.keyset.row8.enum.key7.description=\u00E0
bharosa.authentipad.keypad.german.keyset.row8.enum.key7.value=\u00E0
bharosa.authentipad.keypad.german.keyset.row8.enum.key7.shiftvalue=\u00E0
bharosa.authentipad.keypad.german.keyset.row8.enum.key7.image=kp_v01_00E0.png
bharosa.authentipad.keypad.german.keyset.row8.enum.key7.order=7

bharosa.authentipad.keypad.german.keyset.row8.enum.key8=7
bharosa.authentipad.keypad.german.keyset.row8.enum.key8.name=\u00E1
bharosa.authentipad.keypad.german.keyset.row8.enum.key8.description=\u00E1
bharosa.authentipad.keypad.german.keyset.row8.enum.key8.value=\u00E1
bharosa.authentipad.keypad.german.keyset.row8.enum.key8.shiftvalue=\u00E1
bharosa.authentipad.keypad.german.keyset.row8.enum.key8.image=kp_v01_00E1.png
bharosa.authentipad.keypad.german.keyset.row8.enum.key8.order=8

bharosa.authentipad.keypad.german.keyset.row8.enum.key9=8
bharosa.authentipad.keypad.german.keyset.row8.enum.key9.name=\u00E2
bharosa.authentipad.keypad.german.keyset.row8.enum.key9.description=\u00E2
bharosa.authentipad.keypad.german.keyset.row8.enum.key9.value=\u00E2
bharosa.authentipad.keypad.german.keyset.row8.enum.key9.shiftvalue=\u00E2
bharosa.authentipad.keypad.german.keyset.row8.enum.key9.image=kp_v01_00E2.png
bharosa.authentipad.keypad.german.keyset.row8.enum.key9.order=9

bharosa.authentipad.keypad.german.keyset.row8.enum.key10=9
bharosa.authentipad.keypad.german.keyset.row8.enum.key10.name=\u00E3
bharosa.authentipad.keypad.german.keyset.row8.enum.key10.description=\u00E3
bharosa.authentipad.keypad.german.keyset.row8.enum.key10.value=\u00E3
bharosa.authentipad.keypad.german.keyset.row8.enum.key10.shiftvalue=\u00E3
bharosa.authentipad.keypad.german.keyset.row8.enum.key10.image=kp_v01_00E3.png
bharosa.authentipad.keypad.german.keyset.row8.enum.key10.order=10

bharosa.authentipad.keypad.german.keyset.row8.enum.key11=10
bharosa.authentipad.keypad.german.keyset.row8.enum.key11.name=\u00E4
bharosa.authentipad.keypad.german.keyset.row8.enum.key11.description=\u00E4
bharosa.authentipad.keypad.german.keyset.row8.enum.key11.value=\u00E4
bharosa.authentipad.keypad.german.keyset.row8.enum.key11.shiftvalue=\u00E4
bharosa.authentipad.keypad.german.keyset.row8.enum.key11.image=kp_v01_00E4.png
bharosa.authentipad.keypad.german.keyset.row8.enum.key11.order=11

bharosa.authentipad.keypad.german.keyset.row8.enum.key12=11
bharosa.authentipad.keypad.german.keyset.row8.enum.key12.name=\u00E5
bharosa.authentipad.keypad.german.keyset.row8.enum.key12.description=\u00E5
bharosa.authentipad.keypad.german.keyset.row8.enum.key12.value=\u00E5
bharosa.authentipad.keypad.german.keyset.row8.enum.key12.shiftvalue=\u00E5
bharosa.authentipad.keypad.german.keyset.row8.enum.key12.image=kp_v01_00E5.png
```

```
bharosa.authentipad.keypad.german.keyset.row8.enum.key12.order=12

bharosa.authentipad.keypad.german.keyset.row8.enum.key13=12
bharosa.authentipad.keypad.german.keyset.row8.enum.key13.name=\u00E6
bharosa.authentipad.keypad.german.keyset.row8.enum.key13.description=\u00E6
bharosa.authentipad.keypad.german.keyset.row8.enum.key13.value=\u00E6
bharosa.authentipad.keypad.german.keyset.row8.enum.key13.shiftvalue=\u00E6
bharosa.authentipad.keypad.german.keyset.row8.enum.key13.image=kp_v01_00E6.png
bharosa.authentipad.keypad.german.keyset.row8.enum.key13.order=13



bharosa.authentipad.keypad.german.keyset.row9.enum=German KeyPad Keyset row9
bharosa.authentipad.keypad.german.keyset.row9.enum.key1=0
bharosa.authentipad.keypad.german.keyset.row9.enum.key1.name=\u00E7
bharosa.authentipad.keypad.german.keyset.row9.enum.key1.description=\u00E7
bharosa.authentipad.keypad.german.keyset.row9.enum.key1.value=\u00E7
bharosa.authentipad.keypad.german.keyset.row9.enum.key1.shiftvalue=\u00E7
bharosa.authentipad.keypad.german.keyset.row9.enum.key1.image=kp_v01_00E7.png
bharosa.authentipad.keypad.german.keyset.row9.enum.key1.order=1

bharosa.authentipad.keypad.german.keyset.row9.enum.key2=1
bharosa.authentipad.keypad.german.keyset.row9.enum.key2.name=\u00E8
bharosa.authentipad.keypad.german.keyset.row9.enum.key2.description=\u00E8
bharosa.authentipad.keypad.german.keyset.row9.enum.key2.value=\u00E8
bharosa.authentipad.keypad.german.keyset.row9.enum.key2.shiftvalue=\u00E8
bharosa.authentipad.keypad.german.keyset.row9.enum.key2.image=kp_v01_00E8.png
bharosa.authentipad.keypad.german.keyset.row9.enum.key2.order=2

bharosa.authentipad.keypad.german.keyset.row9.enum.key3=2
bharosa.authentipad.keypad.german.keyset.row9.enum.key3.name=\u00E9
bharosa.authentipad.keypad.german.keyset.row9.enum.key3.description=\u00E9
bharosa.authentipad.keypad.german.keyset.row9.enum.key3.value=\u00E9
bharosa.authentipad.keypad.german.keyset.row9.enum.key3.shiftvalue=\u00E9
bharosa.authentipad.keypad.german.keyset.row9.enum.key3.image=kp_v01_00E9.png
bharosa.authentipad.keypad.german.keyset.row9.enum.key3.order=3

bharosa.authentipad.keypad.german.keyset.row9.enum.key4=3
bharosa.authentipad.keypad.german.keyset.row9.enum.key4.name=\u00EA
bharosa.authentipad.keypad.german.keyset.row9.enum.key4.description=\u00EA
bharosa.authentipad.keypad.german.keyset.row9.enum.key4.value=\u00EA
bharosa.authentipad.keypad.german.keyset.row9.enum.key4.shiftvalue=\u00EA
bharosa.authentipad.keypad.german.keyset.row9.enum.key4.image=kp_v01_00EA.png
bharosa.authentipad.keypad.german.keyset.row9.enum.key4.order=4

bharosa.authentipad.keypad.german.keyset.row9.enum.key5=4
bharosa.authentipad.keypad.german.keyset.row9.enum.key5.name=\u00EB
bharosa.authentipad.keypad.german.keyset.row9.enum.key5.description=\u00EB
bharosa.authentipad.keypad.german.keyset.row9.enum.key5.value=\u00EB
bharosa.authentipad.keypad.german.keyset.row9.enum.key5.shiftvalue=\u00EB
bharosa.authentipad.keypad.german.keyset.row9.enum.key5.image=kp_v01_00EB.png
bharosa.authentipad.keypad.german.keyset.row9.enum.key5.order=5

bharosa.authentipad.keypad.german.keyset.row9.enum.key6=5
bharosa.authentipad.keypad.german.keyset.row9.enum.key6.name=\u00EC
bharosa.authentipad.keypad.german.keyset.row9.enum.key6.description=\u00EC
bharosa.authentipad.keypad.german.keyset.row9.enum.key6.value=\u00EC
bharosa.authentipad.keypad.german.keyset.row9.enum.key6.shiftvalue=\u00EC
bharosa.authentipad.keypad.german.keyset.row9.enum.key6.image=kp_v01_00EC.png
bharosa.authentipad.keypad.german.keyset.row9.enum.key6.order=6
```

```
bharosa.authentipad.keypad.german.keyset.row9.enum.key7=6
bharosa.authentipad.keypad.german.keyset.row9.enum.key7.name=\u00ED
bharosa.authentipad.keypad.german.keyset.row9.enum.key7.description=\u00ED
bharosa.authentipad.keypad.german.keyset.row9.enum.key7.value=\u00ED
bharosa.authentipad.keypad.german.keyset.row9.enum.key7.shiftvalue=\u00ED
bharosa.authentipad.keypad.german.keyset.row9.enum.key7.image=kp_v01_00ED.png
bharosa.authentipad.keypad.german.keyset.row9.enum.key7.order=7

bharosa.authentipad.keypad.german.keyset.row9.enum.key8=7
bharosa.authentipad.keypad.german.keyset.row9.enum.key8.name=\u00EE
bharosa.authentipad.keypad.german.keyset.row9.enum.key8.description=\u00EE
bharosa.authentipad.keypad.german.keyset.row9.enum.key8.value=\u00EE
bharosa.authentipad.keypad.german.keyset.row9.enum.key8.shiftvalue=\u00EE
bharosa.authentipad.keypad.german.keyset.row9.enum.key8.image=kp_v01_00EE.png
bharosa.authentipad.keypad.german.keyset.row9.enum.key8.order=8

bharosa.authentipad.keypad.german.keyset.row9.enum.key9=8
bharosa.authentipad.keypad.german.keyset.row9.enum.key9.name=\u00EF
bharosa.authentipad.keypad.german.keyset.row9.enum.key9.description=\u00EF
bharosa.authentipad.keypad.german.keyset.row9.enum.key9.value=\u00EF
bharosa.authentipad.keypad.german.keyset.row9.enum.key9.shiftvalue=\u00EF
bharosa.authentipad.keypad.german.keyset.row9.enum.key9.image=kp_v01_00EF.png
bharosa.authentipad.keypad.german.keyset.row9.enum.key9.order=9

bharosa.authentipad.keypad.german.keyset.row9.enum.key10=9
bharosa.authentipad.keypad.german.keyset.row9.enum.key10.name=\u00F0
bharosa.authentipad.keypad.german.keyset.row9.enum.key10.description=\u00F0
bharosa.authentipad.keypad.german.keyset.row9.enum.key10.value=\u00F0
bharosa.authentipad.keypad.german.keyset.row9.enum.key10.shiftvalue=\u00F0
bharosa.authentipad.keypad.german.keyset.row9.enum.key10.image=kp_v01_00F0.png
bharosa.authentipad.keypad.german.keyset.row9.enum.key10.order=10

bharosa.authentipad.keypad.german.keyset.row9.enum.key11=10
bharosa.authentipad.keypad.german.keyset.row9.enum.key11.name=\u00F1
bharosa.authentipad.keypad.german.keyset.row9.enum.key11.description=\u00F1
bharosa.authentipad.keypad.german.keyset.row9.enum.key11.value=\u00F1
bharosa.authentipad.keypad.german.keyset.row9.enum.key11.shiftvalue=\u00F1
bharosa.authentipad.keypad.german.keyset.row9.enum.key11.image=kp_v01_00F1.png
bharosa.authentipad.keypad.german.keyset.row9.enum.key11.order=11

bharosa.authentipad.keypad.german.keyset.row9.enum.key12=11
bharosa.authentipad.keypad.german.keyset.row9.enum.key12.name=\u00F2
bharosa.authentipad.keypad.german.keyset.row9.enum.key12.description=\u00F2
bharosa.authentipad.keypad.german.keyset.row9.enum.key12.value=\u00F2
bharosa.authentipad.keypad.german.keyset.row9.enum.key12.shiftvalue=\u00F2
bharosa.authentipad.keypad.german.keyset.row9.enum.key12.image=kp_v01_00F2.png
bharosa.authentipad.keypad.german.keyset.row9.enum.key12.order=12

bharosa.authentipad.keypad.german.keyset.row9.enum.key13=12
bharosa.authentipad.keypad.german.keyset.row9.enum.key13.name=\u00F3
bharosa.authentipad.keypad.german.keyset.row9.enum.key13.description=\u00F3
bharosa.authentipad.keypad.german.keyset.row9.enum.key13.value=\u00F3
bharosa.authentipad.keypad.german.keyset.row9.enum.key13.shiftvalue=\u00F3
bharosa.authentipad.keypad.german.keyset.row9.enum.key13.image=kp_v01_00F3.png
bharosa.authentipad.keypad.german.keyset.row9.enum.key13.order=13



bharosa.authentipad.keypad.german.keyset.row10.enum=German KeyPad Keyset row10
```

```
bharosa.authentipad.keypad.german.keyset.row10.enum.key1=0
bharosa.authentipad.keypad.german.keyset.row10.enum.key1.name=\u00F4
bharosa.authentipad.keypad.german.keyset.row10.enum.key1.description=\u00F4
bharosa.authentipad.keypad.german.keyset.row10.enum.key1.value=\u00F4
bharosa.authentipad.keypad.german.keyset.row10.enum.key1.shiftvalue=\u00F4
bharosa.authentipad.keypad.german.keyset.row10.enum.key1.image=kp_v01_00F4.png
bharosa.authentipad.keypad.german.keyset.row10.enum.key1.order=1

bharosa.authentipad.keypad.german.keyset.row10.enum.key2=1
bharosa.authentipad.keypad.german.keyset.row10.enum.key2.name=\u00F5
bharosa.authentipad.keypad.german.keyset.row10.enum.key2.description=\u00F5
bharosa.authentipad.keypad.german.keyset.row10.enum.key2.value=\u00EF5
bharosa.authentipad.keypad.german.keyset.row10.enum.key2.shiftvalue=\u00F5
bharosa.authentipad.keypad.german.keyset.row10.enum.key2.image=kp_v01_00F5.png
bharosa.authentipad.keypad.german.keyset.row10.enum.key2.order=2

bharosa.authentipad.keypad.german.keyset.row10.enum.key3=2
bharosa.authentipad.keypad.german.keyset.row10.enum.key3.name=\u00F6
bharosa.authentipad.keypad.german.keyset.row10.enum.key3.description=\u00F6
bharosa.authentipad.keypad.german.keyset.row10.enum.key3.value=\u00F6
bharosa.authentipad.keypad.german.keyset.row10.enum.key3.shiftvalue=\u00F6
bharosa.authentipad.keypad.german.keyset.row10.enum.key3.image=kp_v01_00F6.png
bharosa.authentipad.keypad.german.keyset.row10.enum.key3.order=3

bharosa.authentipad.keypad.german.keyset.row10.enum.key4=3
bharosa.authentipad.keypad.german.keyset.row10.enum.key4.name=\u00F7
bharosa.authentipad.keypad.german.keyset.row10.enum.key4.description=\u00F7
bharosa.authentipad.keypad.german.keyset.row10.enum.key4.value=\u00F7
bharosa.authentipad.keypad.german.keyset.row10.enum.key4.shiftvalue=\u00F7
bharosa.authentipad.keypad.german.keyset.row10.enum.key4.image=kp_v01_00F7.png
bharosa.authentipad.keypad.german.keyset.row10.enum.key4.order=4

bharosa.authentipad.keypad.german.keyset.row10.enum.key5=4
bharosa.authentipad.keypad.german.keyset.row10.enum.key5.name=\u00F8
bharosa.authentipad.keypad.german.keyset.row10.enum.key5.description=\u00F8
bharosa.authentipad.keypad.german.keyset.row10.enum.key5.value=\u00F8
bharosa.authentipad.keypad.german.keyset.row10.enum.key5.shiftvalue=\u00F8
bharosa.authentipad.keypad.german.keyset.row10.enum.key5.image=kp_v01_00F8.png
bharosa.authentipad.keypad.german.keyset.row10.enum.key5.order=5

bharosa.authentipad.keypad.german.keyset.row10.enum.key6=5
bharosa.authentipad.keypad.german.keyset.row10.enum.key6.name=\u00F9
bharosa.authentipad.keypad.german.keyset.row10.enum.key6.description=\u00F9
bharosa.authentipad.keypad.german.keyset.row10.enum.key6.value=\u00F9
bharosa.authentipad.keypad.german.keyset.row10.enum.key6.shiftvalue=\u00F9
bharosa.authentipad.keypad.german.keyset.row10.enum.key6.image=kp_v01_00F9.png
bharosa.authentipad.keypad.german.keyset.row10.enum.key6.order=6

bharosa.authentipad.keypad.german.keyset.row10.enum.key7=6
bharosa.authentipad.keypad.german.keyset.row10.enum.key7.name=\u00FA
bharosa.authentipad.keypad.german.keyset.row10.enum.key7.description=\u00FA
bharosa.authentipad.keypad.german.keyset.row10.enum.key7.value=\u00FA
bharosa.authentipad.keypad.german.keyset.row10.enum.key7.shiftvalue=\u00FA
bharosa.authentipad.keypad.german.keyset.row10.enum.key7.image=kp_v01_00FA.png
bharosa.authentipad.keypad.german.keyset.row10.enum.key7.order=7

bharosa.authentipad.keypad.german.keyset.row10.enum.key8=7
bharosa.authentipad.keypad.german.keyset.row10.enum.key8.name=\u00FB
bharosa.authentipad.keypad.german.keyset.row10.enum.key8.description=\u00FB
bharosa.authentipad.keypad.german.keyset.row10.enum.key8.value=\u00FB
```

```
bharosa.authentipad.keypad.german.keyset.row10.enum.key8.shiftvalue=\u00FB
bharosa.authentipad.keypad.german.keyset.row10.enum.key8.image=kp_v01_00FB.png
bharosa.authentipad.keypad.german.keyset.row10.enum.key8.order=8

bharosa.authentipad.keypad.german.keyset.row10.enum.key9=8
bharosa.authentipad.keypad.german.keyset.row10.enum.key9.name=\u00FC
bharosa.authentipad.keypad.german.keyset.row10.enum.key9.description=\u00FC
bharosa.authentipad.keypad.german.keyset.row10.enum.key9.value=\u00FC
bharosa.authentipad.keypad.german.keyset.row10.enum.key9.shiftvalue=\u00FC
bharosa.authentipad.keypad.german.keyset.row10.enum.key9.image=kp_v01_00FC.png
bharosa.authentipad.keypad.german.keyset.row10.enum.key9.order=9

bharosa.authentipad.keypad.german.keyset.row10.enum.key10=9
bharosa.authentipad.keypad.german.keyset.row10.enum.key10.name=\u00FD
bharosa.authentipad.keypad.german.keyset.row10.enum.key10.description=\u00FD
bharosa.authentipad.keypad.german.keyset.row10.enum.key10.value=\u00FD
bharosa.authentipad.keypad.german.keyset.row10.enum.key10.shiftvalue=\u00FD
bharosa.authentipad.keypad.german.keyset.row10.enum.key10.image=kp_v01_00FD.png
bharosa.authentipad.keypad.german.keyset.row10.enum.key10.order=10

bharosa.authentipad.keypad.german.keyset.row10.enum.key11=10
bharosa.authentipad.keypad.german.keyset.row10.enum.key11.name=\u00FE
bharosa.authentipad.keypad.german.keyset.row10.enum.key11.description=\u00FE
bharosa.authentipad.keypad.german.keyset.row10.enum.key11.value=\u00FE
bharosa.authentipad.keypad.german.keyset.row10.enum.key11.shiftvalue=\u00FE
bharosa.authentipad.keypad.german.keyset.row10.enum.key11.image=kp_v01_00FE.png
bharosa.authentipad.keypad.german.keyset.row10.enum.key11.order=11

bharosa.authentipad.keypad.german.keyset.row10.enum.key12=11
bharosa.authentipad.keypad.german.keyset.row10.enum.key12.name=\u00FF
bharosa.authentipad.keypad.german.keyset.row10.enum.key12.description=\u00FF
bharosa.authentipad.keypad.german.keyset.row10.enum.key12.value=\u00FF
bharosa.authentipad.keypad.german.keyset.row10.enum.key12.shiftvalue=\u00FF
bharosa.authentipad.keypad.german.keyset.row10.enum.key12.image=kp_v01_00FF.png
bharosa.authentipad.keypad.german.keyset.row10.enum.key12.order=12

bharosa.authentipad.keypad.german.keyset.row10.enum.key13=12
bharosa.authentipad.keypad.german.keyset.row10.enum.key13.name=\u00FF
bharosa.authentipad.keypad.german.keyset.row10.enum.key13.description=\u00FF
bharosa.authentipad.keypad.german.keyset.row10.enum.key13.value=\u00FF
bharosa.authentipad.keypad.german.keyset.row10.enum.key13.shiftvalue=\u00FF
bharosa.authentipad.keypad.german.keyset.row10.enum.key13.image=kp_v01_00FF.png
bharosa.authentipad.keypad.german.keyset.row10.enum.key13.order=13
```

5. Add frame and key image files to following directories:
   - Key Image Files: `<temp-folder>/WEB-INF/classes/bharosa_properties/alphapad_skins_de`.
   - Frame Image Files: `<temp-folder>//WEB-INF/classes/bharosa_properties/alphapad_bg`.
6. Re-Jar the war using the command:

```
jar -cvfm oracle.oaam.extensions.war <temp-folder>/META-INF/MANIFEST.MF -C
<temp-folder>
```

Note: Make sure original MANIFEST.MF remains same as that contains shared library information.

7. Re-deploy the updated `oracle.oaam.extensions.war` as a shared library with targets as oaam_server and oaam_admin
8. Restart OAAM Servers and validate your changes by accessing application with browser set to German locale.

11

Implementing OTP Anywhere

This chapter explains how to implement OTP Anywhere. OTP Anywhere allows end users to authenticate themselves by entering a server generated one-time-password (OTP). When the OTP is sent via SMS, the user's cell phone serves as a physical second factor that the user has in their possession. As well, the authentication is being sent out-of-band to increase the level of assurance that only the valid user has access to the one-time password.

Benefits of OTP Anywhere are:

- It is built on 11g Challenge Processor framework
- Out of the box integration with Oracle User Messaging Service
- Customizable registration user interface
- Optional Opt-Out functionality
- Email and SMS supported delivery channels

This chapter contains these sections:

11.1 About the Implementation

One-Time Password (OTP) is a form of secondary authentication, which is used in addition to standard user name and password credentials to strengthen the existing authentication and authorization process, thereby providing additional security for users. The application sends a one-time password that is only valid for the current

session to the user. This password is used to challenge the user to verify the user's identity.

Oracle Adaptive Access Manager 11*g* provides the framework to support One Time Password (OTP) authentication using Oracle User Messaging Service (UMS).

This implementation enables an application to use OTP to challenge users with Oracle User Messaging Service (UMS) used as the method to deliver the password.

The high-level integration tasks consist of:

- Prerequisites
- Configuring OTP
- Customizing OTP
- Registering SMS Processor to Perform Work for Challenge Type
- Configuring the Challenge Pads Used for Challenge Types
- Customizing OTP Anywhere Data Storage

11.2 Concepts and Terms

This section provides the terms that are helpful to know as you implement OTP Anywhere.

11.2.1 One Time Password (OTP)

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.

11.2.2 Oracle User Messaging Service (UMS)

The UMS Server orchestrates message flows between applications and users. OAAM uses UMS to send email, SMS, IM, or voice message to the user.

11.2.3 Challenge Processor

A challenge processor is java code that implements the `ChallengeProcessorIntf` interface or extends the `AbstractChallengeProcessor` class. Custom challenge processors can be created to generate a challenge, validate the challenge answer from the user, and check service delivery and availability statuses. By default OAAM has support (or challenge processor implementations) for KBA question challenges and OTP challenges via SMS and email through UMS delivery.

11.2.4 Challenge Type

"Channel" refers to the delivery channel used to send an OTP to the user (Email, SMS, or IM). The challenge type is the channel that OTP is using to challenge the user. A challenge type can be configured for any differences in handling for a challenge that is required. Handling of challenge types could be any specifics for that challenge type, from generating the "secret" used for the challenge to delivering the "secret" to the user

and finally validating the users input. For each type of challenge these primary processes (Generation, Sending, and Validating) could require slightly different code.

11.3 Prerequisites

Ensure that the following prerequisites are met before configuring OTP for your application.

Note: Ensure you are familiar with deploying custom OAAM extensions.

Oracle Adaptive Access Manager is customized through adding customized jars and files to an extensions shared library.

For information, refer to Chapter 7, "Customizing Oracle Adaptive Access Manager.".

11.3.1 Install SOA Suite

Oracle SOA Suite must be installed outside of the OAAM domains. UMS is a part of SOA.

For information, refer to the *Oracle Fusion Middleware Installation Guide for Oracle SOA Suite and Oracle Business Process Management Suite*.

11.3.2 Configure the UMS Driver

UMS must be configured for appropriate delivery gateways on the SOA that the OAAM Server is configured to send messages through.

UMS Drivers connect UMS to the messaging gateways, adapting content to the various protocols supported by UMS. Drivers can be deployed or undeployed independently of one another depending on what messaging channels are available in a given installation.

11.3.2.1 Email Driver

Configure the Email driver to a SMTP server. See the "Configuring the Email Driver" section of *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite* for how to configure the Email driver.

11.3.2.2 SMPP Driver

Short Message Peer-to-Peer (SMPP) is one of the most popular GSM SMS protocols. User Messaging Service includes a prebuilt implementation of the SMPP protocol as a driver that is capable of both sending and receiving short messages.

Note: For SMS, unlike the Email driver that is deployed out-of-the-box, you need to deploy the SMPP driver first before modifying the configurations.

Configure the SMPP driver as described in the "Configuring the SMPP Driver" section of the *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite*. You will need to provide parameter values for connecting to the driver gateway vendor.

Table 11–1 Connecting to the Vendor

Parameter	Description
SmsAccountId	The Account Identifier on the SMS-C. This is your vendor account ID which you need to get from the vendor.
SmsServerHost	The name (or IP address) of the SMS-C server. TransmitterSystemId
TransmitterSystemPassword	The password of the transmitter system. This includes Type of Password (choose from Indirect Password/Create New User, Indirect Password/Use Existing User, and Use Cleartext Password) and Password. This is the password corresponding to your vendor account ID
TransmitterSystemType	The type of transmitter system. The default is Logica.
ReceiverSystemId	The account ID that is used to receive messages. ReceiverSystemPassword
ReceiverSystemType	The type of receiver system. The default is Logica.
ServerTransmitterPort	The TCP port number of the transmitter server.
ServerReceiverPort	The TCP port number of the receiver server.
DefaultEncoding	The default encoding of the SMPP driver. The default is IA5. Choose from the drop-down list: IA5, UCS2, and GSM_DEFAULT.
DefaultSenderAddress	Default sender address

11.4 OTP Setup Overview

OTP using UMS as a delivery method is a standard feature of the OAAM Server. This section contains an overview of the steps required to implement the feature.

Follow the instructions for customizing the OAAM server interface through adding customized jars and files to an extensions shared library. For information, refer to Chapter 7, "Customizing Oracle Adaptive Access Manager.".

Table 11–2 Tasks in the OTP Setup

Task	Description
Configure	Configuration involves Tasks 1 through 3. OTP Challenge is not enabled by default. It has to be enabled by setting these properties.
Task 1- Integrate UMS.	Set up UMS URLs and credentials so that OAAM can communicate with the UMS server.
Task 2 - Make Challenge Types available.	Make it possible for the policies to challenge using OTP via the challenge type.
Task 3 - Enable Registration and User Preferences.	Enable registration and user preferences. The user will use the pages for profile registration and resetting OTP profile.
Customize	Customizations involves Tasks 4 through 7.
Task 4 - Set up the user registration fields and validations.	Set up the registration and preferences page input fields for the user. Input properties includes maximum length for the email address the user can enter, validation for the email address field (expression), and so on. Note: Any user facing strings will need to be duplicated into resource bundle.
Task 5 - Set up Terms and Condition fields.	Additional fields to set up are Terms of Service, Privacy Policy, and so on.

Table 11–2 (Cont.) Tasks in the OTP Setup

Task	Description
Task 6 - Set up registration and challenge page messaging	Customize the messaging that appear on the registration and challenge pages.
Task 7 - Customize OTP message text.	Customize the message containing the One Time Password
Task 8 - Register Processors	The challenge type enum is used to associate a Challenge Type with the java code needed to perform any work related to that challenge type.
Task 9 - Configure challenge pads for challenge types.	Specify the type of device to use based on the purpose of the device.

The UMS OTP implementation is integrated into the OAAM Server login, challenge, and registration flows using the OAAM Server challenge processor framework. For information on the login, challenge, and registration flows, refer to Chapter 2, "Natively Integrating with Oracle Adaptive Access Manager."

11.5 Configuring OTP

This section contains the following topics:

- Integrating UMS
- Enabling OTP Challenge Types
- Enabling Registration and User Preferences

11.5.1 Integrating UMS

The properties to set for the UMS server URLs and credentials are listed below. They can be edited using the Property Editor in OAAM Admin. Note: End point is the Web Services URL that OAAM uses to send calls into UMS.

Table 11–3 UMS Server URLs and Credentials

Property	Default Value	Description
bharosa.uio.default.ums.integration.webservice		UMS Server Webservice URL http://<UMS Server URL>:<UMS Port>/ucs/messaging/webservice
bharosa.uio.default.ums.integration.parlayx.endpoint		UMS Server ParlayX Endpoint URL http://<UMS Server URL>:<UMS Port>/sdpmessaging/parlayx/SendMessageService
bharosa.uio.default.ums.integration.useParlayX	false	Configures the use of webservice or parlayx API. The value is false by default (Webservices recommended)
bharosa.uio.default.ums.integration.userName		Username for UMS server
bharosa.uio.default.ums.integration.password		Password for UMS server
bharosa.uio.default.ums.integtaion.policies		UMS authentication policies
bharosa.uio.default.ums.integration.fromAddress	demo@oracle.com	OAAM from address for OTP messages
bharosa.uio.default.ums.integration.message.status.poll.attempts	3	Number of times to attempt status poll each time the wait page is displayed

Table 11–3 (Cont.) UMS Server URLs and Credentials

Property	Default Value	Description
bharosa.uio.default.ums.integration.message.status.poll.delay	1000	Delay between status polls while the wait page is being displayed
bharosa.uio.default.ums.integration.sleepInterval	10000	
bharosa.uio.default.ums.integration.deliveryPage.delay	3000	

After you set up the UMS server properties, restart the application.

11.5.2 Enabling OTP Challenge Types

Enable challenge types by setting the appropriate property to true. By setting the property to true, policies will be able to challenge using OTP via the challenge type (email, SMS, IM, or Voice). The user will see the email, SMS, IM, or Voice page in registration flow.

The challenge type enum is used to associate a Challenge Type with the java code needed to perform any work related to that challenge type. The Challenge Type ID (ChallengeEmail) should match a rule action returned by the rules when that challenge type is going to be used.

Table 11–4 UMS OTP challenge types

Property	Default Value	Description
bharosa.uio.default.challenge.type.enum.ChallengeEmail.available	false	Availability flag for email challenge type
bharosa.uio.default.challenge.type.enum.ChallengeSMS.available	false	Availability flag for SMS challenge type
bharosa.uio.default.challenge.type.enum.ChallengeIM.available	false	Availability flag for instant message challenge type
bharosa.uio.default.challenge.type.enum.ChallengeVoice.available	false	Availability flag for voice challenge type

11.5.3 Enabling Registration and User Preferences

Enable the registration flow and user preferences by setting these properties to true:

Table 11–5 Enable OTP Profile Registration and Preference Setting

Property	Description
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. User Preferences is a page that allows the user to change their image/phrase, challenge questions, un-register devices, and update their OTP profile.

11.6 Customizing OTP

This section contains the following topics:

- Customizing Registration Fields and Validations

- Customizing Terms and Conditions
- Customizing Registration Page Messaging
- Customizing Challenge Page Messaging
- Customizing OTP Message Text
- Enabling Opt Out Functionality

11.6.1 Customizing Registration Fields and Validations

Mobile registration field definitions and validations for the OTP registration page are shown below.

Add Mobile Input Registration Field Properties to bharosa_server.properties

These properties should be added to bharosa_server.properties.

Table 11–6 Mobile Input - Properties File

Property	Default Value	Description
bharosa.uio.default.userinfo.inputs.enum.mobile	0	Mobile phone enum value
bharosa.uio.default.userinfo.inputs.enum.mobile.name	Mobile Phone	Name for mobile phone field
bharosa.uio.default.userinfo.inputs.enum.mobile.description	Mobile Phone	Description for mobile phone field
bharosa.uio.default.userinfo.inputs.enum.mobile.inputname	cell number	HTML input name for mobile phone field
bharosa.uio.default.userinfo.inputs.enum.mobile.inputtype	text	HTML input type for mobile phone field
bharosa.uio.default.userinfo.inputs.enum.mobile.maxlength	15	HTML input max length for mobile phone field
bharosa.uio.default.userinfo.inputs.enum.mobile.required	true	Required flag for mobile phone field during registration and user preferences
bharosa.uio.default.userinfo.inputs.enum.mobile.order	1	Order on the page for mobile phone field
bharosa.uio.default.userinfo.inputs.enum.mobile.enabled	true	Enabled flag for mobile phone enum item
bharosa.uio.default.userinfo.inputs.enum.mobile.regex	\\D?(\\d{3})\\D?\\D?(\\d{3})\\D?(\\d{4})	Regular expression for validation of mobile phone field
bharosa.uio.default.userinfo.inputs.enum.mobile.errorCode	otp.invalid.mobile	Error code to get error message from if validation of mobile phone entry fails
bharosa.uio.default.userinfo.inputs.enum.mobile.managerClass	com.bharosa.uio.manager.user.DefaultContactInfoManager	Java class to use to save / retrieve mobile phone from data storage

Add Mobile Input Registration Field Properties to client_resource.properties

These properties should be added to the resource bundle.

Table 11–7 Mobile Input - Resource Bundle

Property	Default Value	Description
bharosa.uio.default.userinfo.inputs.enum.mobile.name	Mobile Phone	Name for mobile phone field
bharosa.uio.default.userinfo.inputs.enum.mobile.description	Mobile Phone	Description for mobile phone field

11.6.2 Customizing Terms and Conditions

The following examples show term and conditions definitions for the OTP registration page.

Add Terms and Conditions Definitions to bharosa_server.properties

These properties should be added to bharosa_server.properties.

Table 11–8 Terms and Conditions Checkbox

Property	Default Value	Description
bharosa.uio.default.userinfo.inputs.enum.terms	4	Terms and Conditions enum value
bharosa.uio.default.userinfo.inputs.enum.terms.name	Terms and Conditions	Name for Terms and Conditions checkbox
bharosa.uio.default.userinfo.inputs.enum.terms.description	Terms and Conditions	Description for Terms and Conditions checkbox
bharosa.uio.default.userinfo.inputs.enum.terms.inputname	terms	HTML input name for Terms and Conditions checkbox
bharosa.uio.default.userinfo.inputs.enum.terms.inputtype	checkbox	HTML input type for Terms and Conditions checkbox
bharosa.uio.default.userinfo.inputs.enum.terms.values	true	Required values for Term and Conditions checkbox during registration and user preferences
bharosa.uio.default.userinfo.inputs.enum.terms.maxlength	40	HTML input max length for Terms and Conditions checkbox
bharosa.uio.default.userinfo.inputs.enum.terms.required	true	Required flag for Term and Conditions checkbox during registration and user preferences
bharosa.uio.default.userinfo.inputs.enum.terms.order	5	Order on the page for Terms and Conditions checkbox
bharosa.uio.default.userinfo.inputs.enum.terms.enabled	true	Enabled flag for Terms and Conditions enum item
bharosa.uio.default.userinfo.inputs.enum.terms.regex	.+	Regular expression for validation of Terms and Conditions checkbox
bharosa.uio.default.userinfo.inputs.enum.terms.errorCode	otp.invalid.terms	Error code to get error message from if validation of Terms and Conditions fails
bharosa.uio.default.userinfo.inputs.enum.terms.managerClass	com.bharosa.uio.manager.user.DefaultContactInfoManager	Java class to use to save / retrieve Terms and Conditions from data storage

Add Terms and Conditions Definitions to client_resource.properties

Default messaging for Terms and Conditions is defined by these resource bundle values:

Table 11–9 Messaging of Terms and Conditions

Property	Descriptions
bharosa.uio.default.userinfo.inputs.enum.terms.name	I agree to the [ENTER COMPANY OR SERVICE NAME HERE] terms & conditions. Click to view full Terms & Conditions and Privacy Policy.
bharosa.uio.default.userinfo.inputs.enum.terms.description	Message and Data Rates May Apply. For help or information on this program send "HELP" to [ENTER SHORT/LONG CODE HERE]. To cancel your plan, send "STOP" to [ENTER SHORT/LONG CODE HERE] at anytime. For additional information on this service please go to [ENTER INFORMATIONAL URL HERE]. Supported Carriers: AT&T, Sprint, Nextel, Boost, Verizon Wireless, U.S. Cellular®, T-Mobile®, Cellular One Dobson, Cincinnati Bell, Alltel, Virgin Mobile USA, Cellular South, Unicel, Centennial and Ntelos

The value for `bharosa.uio.default.userinfo.inputs.enum.terms.name` includes placeholder links that use OAAM Server popup messaging for "Terms & Conditions" and "Privacy Policy". The property and resource keys for the contents of the pop-ups are listed as follows.

Table 11–10 Terms & Conditions and Privacy Policy Popup Messaging

Property	Descriptions
bharosa.uio.default.messages.enum.terms.name	Terms and Conditions
bharosa.uio.default.messages.enum.terms.description	PLACEHOLDER TEXT FOR TERMS AND CONDITIONS
bharosa.uio.default.messages.enum.privacy.name	Privacy Policy
bharosa.uio.default.messages.enum.privacy.description	PLACEHOLDER TEXT FOR PRIVACY POLICY

11.6.3 Customizing Registration Page Messaging

Add registration properties to client_resource.properties.

Table 11–11 Registration Resource Bundle

Property	Default Value
bharosa.uio.default.register.userinfo.title	OTP Anywhere Registration
bharosa.uio.default.register.userinfo.message	For your protection please enter your mobile telephone number so we may use it to verify your identity in the future. Please ensure that you have text messaging enabled on your phone.
bharosa.uio.default.register.userinfo.registerdevice.message	Check to register the device that you are currently using as a safe device:
bharosa.uio.default.register.userinfo.continue.button	Continue
bharosa.uio.default.register.userinfo.decline.message	If you decline you will not be asked to register again.
bharosa.uio.default.register.userinfo.decline.button	Decline

Decline Button

To control the presence of the **Decline** button on the profile registration pages, set the following properties:

```
bharosa.uio.default.register.userinfo.decline.enabled = true
```

```
bharosa.uio.default.userpreferences.userinfo.decline.enabled =
true
```

Note: Even if these are true, the button will not show if the Opt Out property is false.

When the **Decline** button is enabled, the user will have another option on the OTP registration page that will allow him to Opt out of OTP challenges. He will not be asked to register OTP again, and will not receive OTP challenges. However, if a Customer Care OTP Profile reset is performed (or reset all) the user will have the opportunity to register OTP again.

Also, even if the user has opted out of OTP, he can access the OTP page in User Preferences and add information and click **Continue**. This will remove the OTP out flag and the user will now be registered for OTP.

11.6.4 Customizing Challenge Page Messaging

Add challenge type fields to client_resource.properties.

Table 11–12 Challenge Type Resource Bundle Items

Property	Default Value
bharosa.uio.default.ChallengeSMS.message	For your protection please enter the code we just sent to your mobile telephone. If you did not receive a code please ensure that text messaging is enabled on your phone and click the resend link below.
bharosa.uio.default.ChallengeSMS.registerdevice.message	Check to register the device that you are currently using as a safe device:
bharosa.uio.default.ChallengeSMS.continue.button	Continue

11.6.5 Customizing OTP Message Text

Add OTP message fields to client_resource.properties.

Table 11–13 Challenge Type Resource Bundle Items

Property	Default Value
bharosa.uio.default.ChallengeSMS.incorrect.message	Incorrect OTP. Please try again.
bharosa.uio.default.ChallengeSMS.message.subject	Oracle OTP Code
bharosa.uio.default.ChallengeSMS.message.body	Your Oracle SMS OTP Code is: {0}

11.6.6 Enabling Opt Out Functionality

This feature is disabled by default. To enable Opt Out for the user, set the property to true.

Table 11–14 OTP opt-out properties

Property	Default Value
bharosa.uio.default.otp.optOut.enabled	false
bharosa.uio.default.otp.optOut.managerClass	com.bharosa.uio.manager.user.DefaultContactInfoManager

11.7 Registering SMS Processor to Perform Work for Challenge Type

The challenge type enum is used to associate a Challenge Type with the java code needed to perform any work related to that challenge type. The Challenge Type ID (ChallengeEmail) should match a rule action returned by the rules when that challenge type is going to be used. "Channel" normally refers to the delivery channel used to send an OTP to the user (Email, SMS, or IM).

Table 11–15 Challenge type enums

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

The properties to register the SMS challenge processor and mark service as available (or unavailable) are listed below.

Table 11–16 Properties to register the SMS challenge processor

Property	Default Value	Description
bharosa.uio.default.challenge.type.enum.ChallengeSMS	2	SMS Challenge enum value
bharosa.uio.default.challenge.type.enum.ChallengeSMS.name	SMS Challenge	Name of SMS challenge type
bharosa.uio.default.challenge.type.enum.ChallengeSMS.description	SMS Challenge	Description of SMS challenge type
bharosa.uio.default.challenge.type.enum.ChallengeSMS.processor	com.bharosa.uio.processor.challenge.ChallengeSMSProcessor	Processor class for SMS challenge type
bharosa.uio.default.challenge.type.enum.ChallengeSMS.requiredInfo	mobile	Required fields to challenge user with SMS challenge type
bharosa.uio.default.challenge.type.enum.ChallengeSMS.available	false	Availability flag for SMS challenge type
bharosa.uio.default.challenge.type.enum.ChallengeSMS.otp	true	OTP flag for SMS challenge type

11.8 Configuring the Challenge Pads Used for Challenge Types

By default, challenge devices that will be used are configured through rules. The rules are under the AuthentiPad checkpoint where you can specify the type of device to use based on the purpose of the device.

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.

Table 11–17 Authentication Device Type

Property	Description
None	No HTML page or authentication pad
DeviceKeyPadFull	Challenge user using KeyPad.
DeviceKeyPadAlpha	Challenge user with the alphanumeric KeyPad (numbers and letters only, no special characters)
DeviceTextPad	Challenge user using TextPad.
DeviceQuestionPad	Challenge user using QuestionPad.
DevicePinPad	Challenge user using PinPad.
DeviceHTMLControl	Challenge user using HTML page instead of an authentication pad.

11.9 Customizing OTP Anywhere Data Storage

This section describes how to customize data storage for OTP Anywhere. You can customize OTP Anywhere by implementing the com.bharosa.uio.manager.user.UserDataManagerIntf interface.

11.9.1 com.bharosa.uio.manager.user.UserDataManagerIntf

The methods used in customization are:

- `public String getUserData(UIOSessionData sessionData, String key);`
- `public void setUserData(UIOSessionData sessionData, String key, String value);`

11.9.2 Default Implementation - com.bharosa.uio.manager.user.DefaultContactInfoManager

The default implementation expands on the interface to break every get and set into two items: **UserDataValue** and **UserDataFlag**. The UserDataFlag is used by OAAM to track that a value has been set, or soft reset a value. When rules are used to check if a user is registered for a given item, the UserDataFlag will be checked in the OAAM database. The UserDataValue is the actual data element entered by the user. In the default implementation this is also stored in the OAAM database, but by extending the `DefaultContactInfoManager` class and overriding the UserDataValue methods (getUserDataValue and setUserDataValue) the data can be stored in an external location if required.

Methods

```
public class DefaultContactInfoManager implements UserDataManagerIntf {

  public String getUserData(UIOSessionData sessionData, String key){
    if (getUserDataFlag(sessionData, key)){
      return getUserDataValue(sessionData, key);
```

```
    }

    return null;
  }

  public void setUserData(UIOSessionData sessionData, String key, String value){
    setUserDataValue(sessionData, key, value);
    setUserDataFlag(sessionData, key, value);
  }

  protected void setUserDataValue(UIOSessionData sessionData, String key, String
value){
    VCryptAuthUser clientUser = sessionData.getClientAuthUser();
    if (clientUser != null) {
      clientUser.setUserData(BharosaConfig.get("oaam.otp.contact.info.prefix",
"otpContactInfo_") + key, value);
    }
  }

  protected String getUserDataValue(UIOSessionData sessionData, String key) {
    VCryptAuthUser clientUser = sessionData.getClientAuthUser();
    if (clientUser != null) {
      return
clientUser.getUserData(BharosaConfig.get("oaam.otp.contact.info.prefix",
"otpContactInfo_") + key);
    }

    return null;
  }


  protected void setUserDataFlag(UIOSessionData sessionData, String key, String
value){
    VCryptAuthUser clientUser = sessionData.getClientAuthUser();
    if (clientUser != null) {
      if (StringUtil.isEmpty(value)) {
clientUser.setUserData(BharosaConfig.get("oaam.otp.contact.info.flag.prefix",
"otpContactInfoFlag_") + key, null);
      } else {
clientUser.setUserData(BharosaConfig.get("oaam.otp.contact.info.flag.prefix",
"otpContactInfoFlag_") + key, "true");
      }
    }
  }

  protected boolean getUserDataFlag(UIOSessionData sessionData, String key) {
    VCryptAuthUser clientUser = sessionData.getClientAuthUser();
    if (clientUser != null) {
      return
Boolean.valueOf(clientUser.getUserData(BharosaConfig.get("oaam.otp.contact.info.fl
ag.prefix", "otpContactInfoFlag_") + key));
    }

    return false;
  }


}
```

11.9.3 Custom Implementation Recommendations

Extend the base implementation class `DefaultContactInfoManager`, and override the "setUserDataValue" and "getUserDataValue" methods to store the data values where appropriate for you implementation.

Leave the default implementation of "setUserDataFlag" and "getUserDataFlag" in place in order for OAAM to properly track which data has been set for the user.

11.9.4 Configuring Properties

OTP Anywhere registration fields are defined by the user defined enum: `bharosa.uio.default.userinfo.inputs.enum`.

Each element has a "managerClass" property that designates which class will be used to store the registration data.

For example, the default mobile phone element is as follows:

```
bharosa.uio.default.userinfo.inputs.enum=Enum for Contact information
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=16
bharosa.uio.default.userinfo.inputs.enum.mobile.required=true
bharosa.uio.default.userinfo.inputs.enum.mobile.order=4
bharosa.uio.default.userinfo.inputs.enum.mobile.enabled=true
bharosa.uio.default.userinfo.inputs.enum.mobile.regex=\\d{1}\\D?(\\d{3})\\D?\\D?(\
\d{3})\\D?(\\d{4})
bharosa.uio.default.userinfo.inputs.enum.mobile.errorCode=otp.invalid.mobile
bharosa.uio.default.userinfo.inputs.enum.mobile.managerClass=com.bharosa.uio.manag
er.user.DefaultContactInfoManager
```

As shown, the default mobile phone definition uses the `DefaultContactInfoManager` class to manage the data. If a custom implementation is desired, the value of the `managerClass` attribute can be updated in OAAM Admin (or through OAAM Extension shared library) to use a custom class.

11.10 Example Configurations

This section contains the following topics:

- Additional Registration Field Definitions Examples
- Additional Challenge Message Examples
- Additional Processors Registration Examples

11.10.1 Additional Registration Field Definitions Examples

Additional registration field definitions are shown below.

Table 11–18 Contact Information Inputs

Property	Description
inputname	Name used for the input field in the HTML form
inputtype	Set for text or password input
maxlength	Maximum length of user input
required	Set if the field is required on the registration page
order	The order displayed in the user interface
regex	Regular expression used to validate user input for this field
errorCode	Error code used to look up validation error message (bharosa.uio.<application ID>.error.<errorCode>)
managerClass	java class that implements com.bharosa.uio.manager.user.UserDataManagerIntf (if data is to be stored in Oracle Adaptive Access Manager database this property should be set to com.bharosa.uio.manager.user.DefaultContactInfoManager)

11.10.1.1 Email Input

The following is an example of an enum defining email registration on the OTP registration page of an authenticator:

Table 11–19 Email Input

Property	Default Value	Description
bharosa.uio.default.userinfo.inputs.enum.email	1	Email address enum value
bharosa.uio.default.userinfo.inputs.enum.email.name	Email Address	Name for email address field
bharosa.uio.default.userinfo.inputs.enum.email.description	Email Address	Description for email address field
bharosa.uio.default.userinfo.inputs.enum.email.inputname	email	HTML input name for email address field
bharosa.uio.default.userinfo.inputs.enum.email.inputtype	text	HTML input type for email address field
bharosa.uio.default.userinfo.inputs.enum.email.maxlength	40	HTML input max length for email address field
bharosa.uio.default.userinfo.inputs.enum.email.required	true	Required flag for email address field during registration and user preferences
bharosa.uio.default.userinfo.inputs.enum.email.order	2	Order on the page for email address field
bharosa.uio.default.userinfo.inputs.enum.email.enabled	false	Enabled flag for email address enum item
bharosa.uio.default.userinfo.inputs.enum.email.regex	.+@[a-zA-Z_]+?\\.[a-zA-Z]{2,3}	Regular expression for validation of email address field
bharosa.uio.default.userinfo.inputs.enum.email.errorCode	otp.invalid.email	Error code to get error message from if validation of email address entry fails
bharosa.uio.default.userinfo.inputs.enum.email.managerClass	com.bharosa.uio.manager.user.DefaultContactInfoManager	Java class to use to save / retrieve email address from data storage

11.10.1.2 Phone Input

The following is an example of an enum defining phone registration on the OTP registration page of an authenticator:

Table 11–20 Phone Input

Property	Default Value	Description
bharosa.uio.default.userinfo.inputs.enum.phone	2	Phone number enum value
bharosa.uio.default.userinfo.inputs.enum.phone.name	Phone Number	Name for phone number field
bharosa.uio.default.userinfo.inputs.enum.phone.description	Phone Number	Description for phone number field
bharosa.uio.default.userinfo.inputs.enum.phone.inputname	phone	HTML input name for phone number field
bharosa.uio.default.userinfo.inputs.enum.phone.inputtype	text	HTML input type for phone number field
bharosa.uio.default.userinfo.inputs.enum.phone.maxlength	15	HTML input max length for phone number field
bharosa.uio.default.userinfo.inputs.enum.phone.required	true	Required flag for phone number field during registration and user preferences
bharosa.uio.default.userinfo.inputs.enum.phone.order	3	Order on the page for phone number field
bharosa.uio.default.userinfo.inputs.enum.phone.enabled	false	Enabled flag for phone number enum item
bharosa.uio.default.userinfo.inputs.enum.phone.regex	\\D?(\\d{3})\\D?\\D?(\\d{3})\\D?(\\d{4})	Regular expression for validation of phone number field
bharosa.uio.default.userinfo.inputs.enum.phone.errorCode	otp.invalid.phone	Error code to get error message from if validation of phone number entry fails
bharosa.uio.default.userinfo.inputs.enum.phone.managerClass	com.bharosa.uio.manager.user.DefaultContactInfoManager	Java class to use to save / retrieve phone number from data storage

11.10.1.3 IM Input

The following is an example of an enum defining IM registration on the OTP registration page of an authenticator:

Table 11–21 IM Input

Property	Default Value	Description
bharosa.uio.default.userinfo.inputs.enum.im	3	Instant message enum value
bharosa.uio.default.userinfo.inputs.enum.im.name	Instant Messaging	Name for instant message field
bharosa.uio.default.userinfo.inputs.enum.im.description	Instant Messaging	Description for instant message field
bharosa.uio.default.userinfo.inputs.enum.im.inputname	im	HTML input name for instant message field
bharosa.uio.default.userinfo.inputs.enum.im.inputtype	text	HTML input type for instant message field
bharosa.uio.default.userinfo.inputs.enum.im.maxlength	15	HTML input max length for instant message field
bharosa.uio.default.userinfo.inputs.enum.im.required	true	Required flag for instant message field during registration and user preferences
bharosa.uio.default.userinfo.inputs.enum.im.order	4	Order on the page for instant message field
bharosa.uio.default.userinfo.inputs.enum.im.enabled	false	Enabled flag for instant message enum item

Table 11–21 (Cont.) IM Input

Property	Default Value	Description
bharosa.uio.default.userinfo.inputs.enum.im.regex	TBD	Regular expression for validation of instant message field
bharosa.uio.default.userinfo.inputs.enum.im.errorCode	otp.invalid.im	Error code to get error message from if validation of instant message entry fails
bharosa.uio.default.userinfo.inputs.enum.im.managerClass	com.bharosa.uio.manager.user.DefaultContactInfoManager	Java class to use to save / retrieve instant message from data storage

11.10.2 Additional Challenge Message Examples

Other examples of challenge message resource bundles are shown below.

11.10.2.1 Customize OTP Email Message

OTP Email message properties are shown below.

Table 11–22 Customize OTP Email Message

Property	Default Value	Description
bharosa.uio.default.ChallengeEmail.message.from.name	Oracle ASA Test	Email message from address
bharosa.uio.default.ChallengeEmail.message.subject	Oracle OTP Code	Email message subject
bharosa.uio.default.ChallengeEmail.message.body	Your Oracle Email OTP Code is: {0}	Email message body

11.10.2.2 Customize OTP IM Message

OTP IM message properties are shown below.

Table 11–23 Customize OTP IM Message

Property	Default Value	Description
bharosa.uio.default.ChallengeIM.message.from.name	Oracle ASA Test	IM message from name
bharosa.uio.default.ChallengeIM.message.subject	Oracle OTP Code	IM message subject
bharosa.uio.default.ChallengeIM.message.body	Your Oracle IM OTP Code is: {0}	IM message body

11.10.2.3 Customize OTP Voice Message

OTP Voice message properties are shown below.

Table 11–24 Customize OTP Voice Message

Property	Default Value	Description
bharosa.uio.default.ChallengeVoice.message.subject	Oracle OTP Code	Voice message subject
bharosa.uio.default.ChallengeVoice.message.body	Your Oracle Voice OTP Code is: {0}	Voice message body

11.10.3 Additional Processors Registration Examples

Additional processor registration properties are listed below.

Table 11–25 Challenge type enums

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

11.10.3.1 Register Email Challenge Processor

The properties to register the email challenge processor and mark service as available (or unavailable) are listed below.

Table 11–26 Properties to register the email challenge processor

Property	Default Value	Description
bharosa.uio.default.challenge.type.enum.ChallengeEmail	1	Email Challenge enum value
bharosa.uio.default.challenge.type.enum.ChallengeEmail.name	Email Challenge	Name of email challenge type
bharosa.uio.default.challenge.type.enum.ChallengeEmail.description	Email Challenge	Description of email challenge type
bharosa.uio.default.challenge.type.enum.ChallengeEmail.processor	com.bharosa.uio.processor.challenge.ChallengeEmailProcessor	Processor class for email challenge type
bharosa.uio.default.challenge.type.enum.ChallengeEmail.requiredInfo	email	Required fields to challenge user with email challenge type
bharosa.uio.default.challenge.type.enum.ChallengeEmail.available	false	Availability flag for email challenge type
bharosa.uio.default.challenge.type.enum.ChallengeEmail.otp	true	OTP flag for email challenge type

11.10.3.2 Register IM Challenge Processor

The properties to register the IM challenge processor and mark service as available (or unavailable) are listed below.

Table 11–27 Properties to register the IM challenge processor

Property	Default Value	Description
bharosa.uio.default.challenge.type.enum.ChallengeIM	3	Instant message Challenge enum value
bharosa.uio.default.challenge.type.enum.ChallengeIM.name	IM Challenge	Name of instant message challenge type
bharosa.uio.default.challenge.type.enum.ChallengeIM.description	Instant Message Challenge	Description of instant message challenge type
bharosa.uio.default.challenge.type.enum.ChallengeIM.processor	com.bharosa.uio.processor.challenge.ChallengeIMProcessor	Processor class for instant message challenge type
bharosa.uio.default.challenge.type.enum.ChallengeIM.requiredInfo	mobile	Required fields to challenge user with instant message challenge type
bharosa.uio.default.challenge.type.enum.ChallengeIM.available	false	Availability flag for instant message challenge type
bharosa.uio.default.challenge.type.enum.ChallengeIM.otp	true	OTP flag for instant message challenge type

11.10.3.3 Register Voice Challenge Processor

The properties to register the Voice challenge processor and mark service as available (or unavailable) are listed below.

Table 11–28 Properties to register the Voice challenge processor

Property	Default Value	Description
bharosa.uio.default.challenge.type.enum.ChallengeVoice	4	Voice Challenge enum value
bharosa.uio.default.challenge.type.enum.ChallengeVoice.name	Voice Challenge	Name of voice challenge type
bharosa.uio.default.challenge.type.enum.ChallengeVoice.description	Voice Challenge	Description of voice challenge type
bharosa.uio.default.challenge.type.enum.ChallengeVoice.processor	com.bharosa.uio.processor.challenge.ChallengeVoiceProcessor	Processor class for voice challenge type
bharosa.uio.default.challenge.type.enum.ChallengeVoice.requiredInfo	phone	Required fields to challenge user with voice challenge type
bharosa.uio.default.challenge.type.enum.ChallengeVoice.available	false	Availability flag for voice challenge type
bharosa.uio.default.challenge.type.enum.ChallengeVoice.otp	true	OTP flag for voice challenge type

11.11 Challenge Use Case

An example challenge scenario is presented below.

1. Oracle Adaptive Access Manager Server presents the user with the user name page.
2. The user submits his user name on the user name page.
3. Oracle Adaptive Access Manager fingerprints the user device and runs pre-authentication rules to determine if the user should be allowed to proceed to the password page.
4. The user is allowed to proceed to the password page and he enters his password.
5. The OAAM policies indicate that the user should be challenged.
6. The challenge checkpoint is run to determine the type of challenge to use (KBA, Email, SMS, and so on). If SMS challenge is returned, the SMS Challenge Processor is loaded and used to generate and deliver an OTP to the user via SMS.
7. Once the SMS has been sent, the user is presented with a challenge page indicating that his OTP has been sent to him in an SMS.
8. User submits correct OTP to continue into application and complete the login flow.

The OTP generated and sent to the user is only valid for one correct submission within a single HTTP session. If the user's HTTP session expires and a new OTP will be generated and sent if he is challenged again in a later session.

12

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.

12.1 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 4.1.3, "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*.

12.2 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`.

12.3 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.
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

12.4 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 occcurred while executing dynamic
action", e);
                }

                //Write appropriate asserts to check if the configurable action
has executed properly
        }

    public void tearDown() throws Exception {

    }
}
```

13

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.devideId` 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 7, "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.

14 Extending Device Identification

This chapter describes how to extend device identification in a typical deployment. It includes the following topics:

14.1 When to Use Extend Device Identification

For most typical deployments, the out-of-the-box device identification satisfies client requirements. Out-of-the-box device identification uses data from the browser and OAAM flash movie. The following are the typical scenarios when you could consider extending device identification:

- The OAAM flash movie cannot be used to obtain client details as the client side browser does not support Flash (example: iPhone, iPad, and so on)
- There is a need to extract stronger device identification data from the client using a non-flash plug-in that can run inside the browser

14.2 Prerequisites

The prerequisites for performing tasks to extend device identification in Oracle Adaptive Access Manager are provided in the following list:

- You have knowledge of Java programming language since a custom device identification plug-in has to be developed using Java.
- You have determined what pieces of information about client device have to be collected and what technology will be used to collect that. Typical technologies you can consider are applets, JavaScript, and so on.
- You understand the process of developing and deploying the OAAM Extensions Shared Library.

14.3 Developing a Custom Device Identification Plug-in

The custom device identification plug-in is software that extends the out-of-the-box device identification provided by Oracle Adaptive Access Manager.

14.3.1 Implement the Client Side Plug-in

Implement the client side plug-in that can run in the client browser. This involves coding the client side plug-in using the appropriate technology.

The client side plug-in should satisfy the following requirements:

- It can run on the client side browser without altering the web page. It is invisible and does not alter user control flow.
- It can communicate with OAAM Server and post data using the HTTP protocol.
- **Very Important**: It can use the existing OAAM "HTTP Session" while posting the data. This is very important for the device identification to work properly.
- The list of data/values that are collected by the plug-in uniquely identifies a client device.
- The plug-in can retrieve and store a cookie equivalent on the client machine.
- Plug-in can submit the following parameters to flash`Fingerprint.do` URL on OAAM Server using HTTP Post:

Table 14–1 Parameters to flashFingerprint.do URL

Name of the parameter	Description
client	Name of the client plug-in. A constant value that indicates the plug-in type.
fp	Concatenated string that has all the name-value pairs that identify the client side. Name-value pairs is concatenated using "&" and name-value is separated using "=". Example: If os_name and os_version are collected by plug-in then the fp string value looks like "os_name=windows&os_version=7
<as determined by the implementation>	Send the cookie equivalent value stored/maintained by the client plug-in.

14.3.2 Add Properties related to Custom Device Identification Plug-in to OAAM Extensions Shared Library

Add the following properties as enum element to vcrypt.fingerprint.type.enum to bharosa_server.properties of the OAAM Extensions Shared Library war.

Note: Replace <plug-in-name> with a string that represents your plug-in. Do not use the strings 'flash', 'browser' as they are already used by the OAAM product.

Table 14–2 vcrypt.fingerprint.type.enum elements

Property Name	Value Description
vcrypt.fingerprint.type.enum.<plugin-name>	Integer value above 100
vcrypt.fingerprint.type.enum. <plug-in-name>.name	Name that represents the plug-in
vcrypt.fingerprint.type.enum. <plug-in-name>.description	Description of the plug-in

Table 14–2 (Cont.) vcrypt.fingerprint.type.enum elements

Property Name	Value Description
vcrypt.fingerprint.type.enum. <plug-in-name>.processor	Fully qualified java class name of the processor class that implements device identification logic on the server side. See next section for details on how to implement this class.
vcrypt.fingerprint.type.enum. <plug-in-name>.header_list	Comma separated list of data that is collected by the client side plug-in.
vcrypt.fingerprint.type.enum. <plug-in-name>.header_name_nv	Comma separated list of data and readable name of those data.
vcrypt.fingerprint.type.enum. <plug-in-name>. header_value_nv	Comma separated list of mappings of value to readable string of those values
bharosa.uio.default.device.identification.scheme	<plug-in-name> Note: This is very important for OAAM to use the custom device identification

14.3.3 Extend/Implement the DeviceIdentification Plug-in class

Extend the `DeviceIdentification` plug-in class: `com.bharosa.uio.processor.device.DeviceIdentificationProcessorBase` and implement the following methods:

14.3.3.1 getPlugInHTML

```
public String getPlugInHTML();
```

Implementation should return a valid plug-in HTML that can be embedded into login pages. The HTML should take care of handling exceptions like if the supporting technology is not available or disabled on the client.

An example for plug-in HTML is shown below:

```
<applet alt="Browser has Java disabled" name="OAAMDeviceIdentifier" width="0"
height="0"
        code="com.bharosa.uio.processor.device.SampleAppletDeviceIdentifierClient"
        codebase="applet"
        archive="oaam_device_sample_applet.jar">
</applet>
```

Note: This method is called by the oaamLoginPage.jsp when the user navigates to login page.

14.3.4 getFingerPrint

```
public String getFingerPrint();
```

This method should implement logic that creates a unique fingerprint that identifies the client device using the data sent by the plug-in.

This method is called when the client side plug-in submits device identification data to OAAM Server.

This method should call the `UIOContext.getCurrentInstance().getRequest` to get handle to `HttpServletRequest` object to read the data sent by the client plug-in.

As mentioned in the previous section, client plug-in would send list of data points as single string as the value of "fp" request parameter.

This class should "tokenize" this string to determine the list of datapoints and their values.

14.3.5 getDigitalCookie

```
public String getDigitalCookie();
```

Implementation should return the digital cookie sent by the client plug-in. It is the responsibility of the client and server to designate an `Http` parameter that indicates the digital cookie.

This method should call the `UIOContext.getCurrentInstance().getRequest` to get handle to `HttpServletRequest` object to read the data sent by the client plug-in.

14.3.6 getClientDataMap

```
public Map getClientDataMap(HttpServletRequest request);
```

Implementation should read the data from request and store it into a map that can be used for logging or auditing purposes.

14.4 Overview of Interactions

Following is the overview of how the device identification plug-in works and interacts with OAAM Server:

1. The user navigates to the OAAM user login page on the OAAM Server.
2. The OAAM Server uses the device identification configuration and appropriately instantiates the device identification plug-in class. It then asks the plug-in class for the HTML that needs to be embedded in the user login page. The OAAM Server returns the user login page with the device identification plug-in HTML.
3. Once the login page is rendered, the client based plug-in is activated and collects information about the device.
4. The client plug-in then submits the collected data to the device identification URL on the OAAM Server.
5. The OAAM Server then calls the device identification plug-in to obtain the fingerprint based on collected data from the client plug-in.
6. It then checks if the fingerprint corresponds to an existing device. If not, then it creates a new device and associates the fingerprint to that device.
7. The OAAM Server then calls the device identification plug-in to get the digital cookie. If digital cookie does not exist then a new one is created.
8. The digital cookie is returned to the client plug-in so that it is stored on the client machine.
9. Once the User ID is entered, using the digital cookie or browser cookie or both, the user request is associated to the device.
10. After the authentication (success/failure), the user request is updated with the authentication result.

11. If the same device is used for future logins, the digital cookie can be used to look up the device without having to fingerprint.

14.5 Compile, Assemble and Deploy

Compile the custom device identification plug-in class and assemble the OAAM Extensions Shared library. Refer to Chapter 7, "Customizing Oracle Adaptive Access Manager" for instructions.

14.6 Important Note About Implementing the Plug-In

When implementing the plug-in, keep the following points in mind:

- Make sure the custom device identification class outputs a valid HTML required to activate the client side plug-in.
- Make sure the client side plug-in posts the data to OAAM Server using the "existing HTTP Session".

15

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.

15.1 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.

15.2 Definitions of Variables and Parameters

Table 15–1 lists the parameter and response variable in the interaction between the flash movie and the application.

Table 15–1 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.

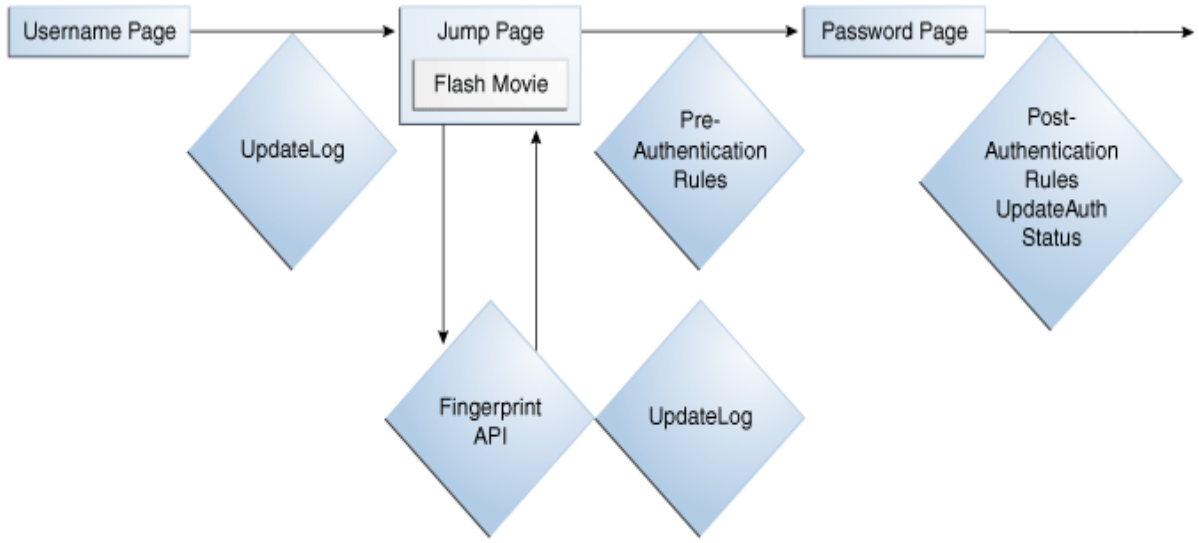
15.3 Option 1

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.

15.3.1 Option 1 Flow

Figure 15–1 shows the flow of Option 1.

Figure 15–1 Option 1



1. The user is presented with the user name page
2. The user submits the user name
   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

15.3.2 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.setAttrubute("cookieSet");
return cookieSet.getFlashCookie();
    } catch (Exception e) {
 // Handle fingerprinting error
    }
    return "";
} // flashFingerPrint
```

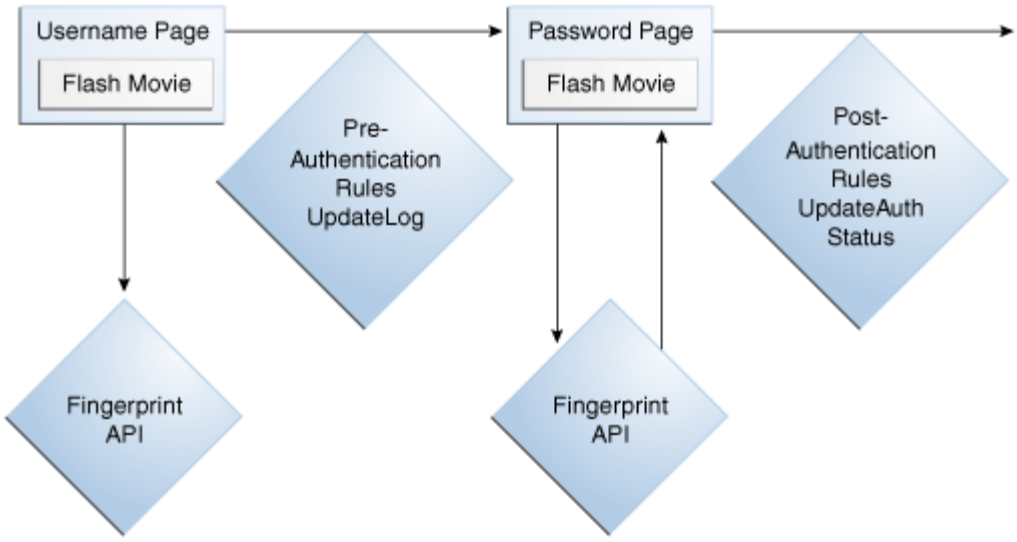
15.4 Option 2

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 user name page and the password page.

15.4.1 Option 2 Flow

Figure 15–2 shows the flow of Option 2.

Figure 15–2 Option 2



1. The user is presented with the user name 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 user name
   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

15.4.2 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
```

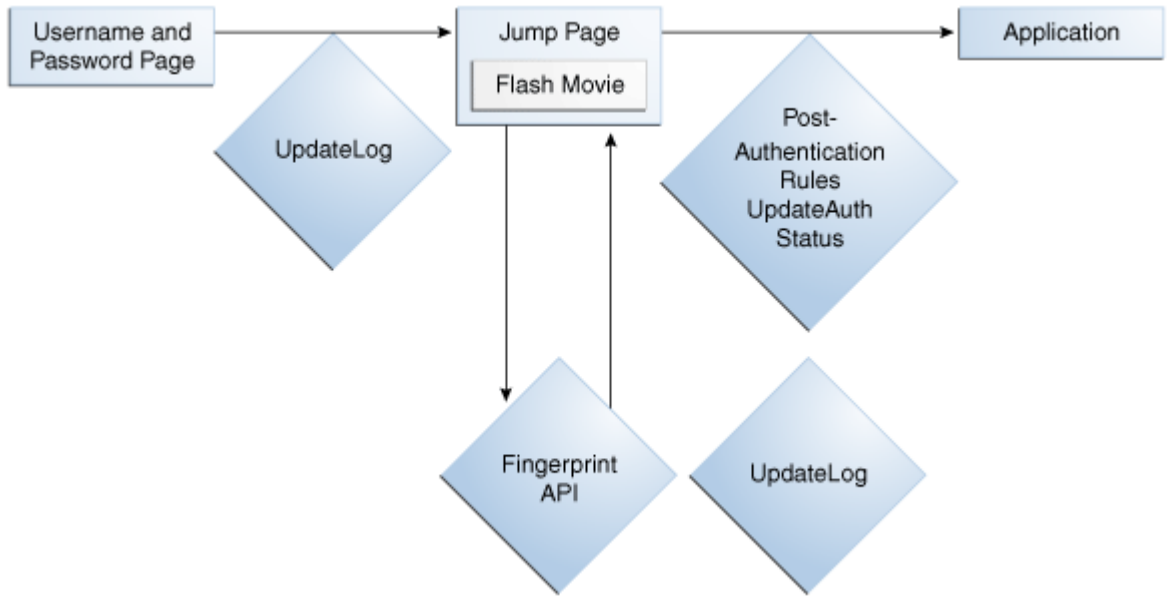
15.5 Option 3

Option 3 is an implementation using a single page for user name 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.

15.5.1 Option 3 Flow

Figure 15–3 shows the flow of Option 3.

Figure 15–3 Option 3 Flow



1. The user is presented with a single user name and password page
2. The user submits the user name 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

15.5.2 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.setAttrubute("cookieSet");
               return cookieSet.getFlashCookie();
    } catch (Exception e) {
 // Handle fingerprinting error
    }
    return "";
} // flashFingerPrint
```

15.6 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(tru);

              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.flashFPType.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;
        }
```

Part IV

Authentication and Password Management Integration

Part IV contains a chapter on Oracle Adaptive Access Manager, Oracle Access Manager, and Oracle Identity Manager integration.

16

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 Administrator's Guide for Oracle Access Manager with Oracle Security Token Service*.

16.1 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

16.2 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 V

Migration and Lifecycle Management

Part V contains the following chapters:

17

Migrating Native Applications to OAAM 11*g*

This chapter covers the tasks involved in migrating an existing natively integrated 10.1.4.5 application that is currently using SOAP authentication to 11*g*.

17.1 Preparing for Migration

Pre-requisites are as follows for migration of your existing natively integrated application:

- Client should be using OAAM Shared Library for Native Integration using SOAP
- Client should specify the configurable properties in `bharosa_server.properties` and this file should be in the Java Classpath of the client application
- See Section 17.4, "Migrating Native Applications that Cannot Use OAAM Shared Library" if the Native Application cannot use the OAAM Shared Library

17.2 Migrating Native Static Linked (In Proc) Applications to OAAM 11*g*

This native integration involves only local API calls and therefore no remote server risk engine calls. The integration embeds the processing engine for OAAM with the application and enables it to leverage the underlying database directly for processing.

To migrate the natively integrated inproc application to OAAM 11*g*, proceed as follows:

17.2.1 Use the OAAM Shared Library Instead of Static Linking to OAAM Jars

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>
```

17.2.2 Move All Configurable Properties into bharosa_server.properties File

As part of migrating the application, you must perform these steps:

1. Move all the configurable properties to `bharosa_server.properties`.
2. Remove/delete all other OAAM property files from the native application.

3. Remove/delete all old OAAM jar files.

17.3 Migrating Native SOAP Applications to OAAM 11*g*

The web application communicates with OAAM via Web Services.

Follow the procedures in this section to migrate your native SOAP application to OAAM 11*g*.

17.3.1 Use OAAM Shared Library Instead of Static Linking to OAAM Jars

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>
```

17.3.2 Move All Configurable Properties into the bharosa_server.properties File

As part of migrating the application, you must perform these steps:

1. Move all the configurable properties to `bharosa_server.properties`.
2. Make sure the following properties are set in `bharosa_server.properties`:
   - `vcrypt.tracker.soap.useSOAPServer=true`
   - `vcrypt.soap.disable=false`
   - `bharosa.config.impl.classname=com.bharosa.common.util.BharosaConfigPropsImpl`
   - `bharosa.config.load.impl.classname=com.bharosa.common.util.BharosaConfigLoadPropsImpl`
3. Remove/delete all other OAAM property files from the native application
4. Remove/delete all old OAAM jar files

17.3.3 Configure SOAP/WebServices Access

For details on configuring SOAP/WebServices Access, refer to "Configuring SOAP Web Services Access" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*.

17.4 Migrating Native Applications that Cannot Use OAAM Shared Library

The process below covers migrating your existing 10.1.4.5 Natively Integrated application that is currently using SOAP authentication to 11*g*.

17.4.1 Use the OAAM 11*g* Jar Files

After those files are copied, you can copy the `oaam_core.jar` file from the *$ORACLE_HOME*`/oaam/cli/lib` folder into your applications library directory. *$ORACLE_HOME* is usually the `ORACLE_IDM1` folder in the Middleware Home.

17.4.2 Copy the OAAM 11*g* Property Files

All updated property files and libraries are located in the *$ORACLE_HOME*/oaam/cli folder. The conf/bharosa_properties folder contains the updated properties, and the lib folder contains the updated libraries.

To upgrade your existing natively integrated application, you can start by removing the contents of your existing bharosa_properties folder, and replacing them with the contents of the *$ORACLE_HOME*/oaam/cli/conf/bharosa_properties directory.

17.4.3 Specify the Configurable Properties in the bharosa_server.properties File

In 10*g* all client specific configuration overrides were created in the bharosa_client.properties file, now those overrides need to be created in the bharosa_server.properties file. This was typically the file modified on the server side for the same purpose. A bharosa_server.properties file that contains the contents of your old bharosa_client.properties with the addition of the following new properties needs to be created in your application's bharosa_properties folder that contains the following information:

```
# New Properties

vcrypt.tracker.soap.useSOAPServer=true

vcrypt.soap.disable=false

bharosa.config.impl.classname=com.bharosa.common.util.BharosaConfigPropsImpl

bharosa.config.load.impl.classname=com.bharosa.common.util.BharosaConfigLoadPropsImpl
```

These new properties will tell the new libraries to use the Generic SOAP implementation classes for communicating with the OAAM Server component, and instead of looking to the OAAM database to read the properties typically retrieved from the BharosaConfig class to retrieve them from the local property files.

It is noted above that these properties are to be used in addition to the existing contents of your bharosa_client.properties file which should include your soap user name, and soap keystore information. Note: If you did not have SOAP authentication setup in 10*g*, you will need to refer to "Setting Up Encryption" in the 10.1.4.5 *Oracle Adaptive Access Manager Installation and Configuration Guide* for creating a SOAP keystore for use with the new 11*g* environment.

18

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

18.1 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 18–1 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.

18.2 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 18–2 Configuring Oracle Identity Manager Property Values

Property Name	Property Values
oaam.oim.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*.

5. Restart the Oracle Adaptive Access Manager server-related managed server.

18.3 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 with Oracle Security Token Service*.

18.4 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:

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 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 18–3 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.

18.5 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 user name/password.

18.6 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*.

18.7 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 VI

Custom Development

Part VI contains the following chapter:

- Chapter 19, "Creating OAAM Oracle BI Publisher Reports"
- Chapter 20, "Developing Custom Challenge Processors"
- Chapter 21, "Creating a View of a Non-OAAM Database"
- Chapter 22, "Developing a Custom Loader for OAAM Offline"

19

Creating OAAM Oracle BI Publisher Reports

This chapter contains instructions on creating Oracle BI Publisher reports on data in the OAAM schema.

19.1 Create Oracle BI Publisher Reports on Data in the OAAM Schema

Refer to the following sections to create OAAM reports from the Oracle Adaptive Access Manager database. In code listings OAAM table and field names are bold and italic.

19.1.1 Create a Data Model

Refer to the instructions in "Creating a New Report" at the following URL:

http://download.oracle.com/docs/cd/E12844_01/doc/bip.1013/e12187/T518230T518233.htm

This section is from the *Oracle Business Intelligence Publisher Report Designer's Guide* at the following URL:

http://download.oracle.com/docs/cd/E12844_01/doc/bip.1013/e12187/toc.htm

19.1.2 Map User Defined Enum Numeric Type Codes to Readable Names

Several fields in many tables are numeric type codes, which correspond to OAAM User Defined Enums. Refer to Chapter 7, "Customizing Oracle Adaptive Access Manager" for more information about OAAM User Defined Enums. Information on how to map those type codes to readable names is presented in this section.

There are two methods for resolving these names, and the one to choose depends on whether you need to display English only or you need to display internationalized strings.

19.1.2.1 Results Display

To display a readable string rather than a type code value in the report output, the report writer will need to add a join to the tables that hold the User Defined Enums, and then add the field to the select clause.

19.1.2.2 English Only User Defined Enum Result Display

The following SQL code shows how to add the join criteria to the query:

```
SELECT …
FROM …
```

```
LEFT OUTER JOIN (
      SELECT enumElement.num_value, enumElement.label
      FROM v_b_enum enum
            INNER JOIN v_b_enum_elmnt enumElement ON on enum.enum_id = enum_
element.enum_id
      WHERE enum.prop_name = 'enum name') alias
      ON table.type_field = alias.num_value
…
```

In this code, `table.type_field` is the field containing a type code value that you want to replace with a string. Alias is the name you are giving the inner select clause. Finally, `enum_name` is the property name of the User Defined Enum.

To display in the report, you need to add *alias*.label to the select clause.

19.1.2.3 Internationalized User Defined Enum Result Display

The following SQL code shows how to add the join criteria to the query:

```
SELECT …
FROM …
LEFT OUTER JOIN (
      SELECT t0.config_value, element.num_value
      FROM v_b_config_rb t0
      INNER JOIN (
            SELECT enum_element.num_value, enum_element.str_value, enum.prop_name
            FROM v_b_enum enum
                  INNER JOIN v_b_enum_elmnt enum_element ON enum.enum_id = enum_
element.enum_id
            WHERE enum.prop_name = 'enum name') element
            ON t0.config_name=element.prop_name || '.' || element.str_value ||
'.name'
      WHERE t0.locale_id = (
            SELECT locale_id FROM v_b_locale
            WHERE language = substr(:xdo_user_ui_locale, 1, 2)
                  AND country = substr(:xdo_user_ui_locale, 4, 2)
                  AND (substr(:xdo_user_ui_locale, 1, 2) in ('de', 'en', 'es',
'fr', 'it', 'ja', 'ko')
                        OR (substr(:xdo_user_ui_locale, 1, 2) = 'pt' AND
substr(:xdo_user_ui_locale, 4, 2) = 'BR')
                        OR (substr(:xdo_user_ui_locale, 1, 2) = 'zh' AND
substr(:xdo_user_ui_locale, 4, 2) IN ('CN', 'TW')))
            UNION SELECT locale_id FROM v_b_locale
            WHERE language = substr(:xdo_user_ui_locale, 1, 2)
                  AND NOT EXISTS(SELECT locale_id FROM v_b_locale
                  WHERE language = substr(:xdo_user_ui_locale, 1, 2)
                        AND country = substr(:xdo_user_ui_locale, 4, 2))
                        AND country IS NULL
                        AND (substr(:xdo_user_ui_locale, 1, 2) in ('de', 'en',
'es', 'fr', 'it', 'ja', 'ko')
                              OR (substr(:xdo_user_ui_locale, 1, 2) = 'pt' AND
substr(:xdo_user_ui_locale, 4, 2) = 'BR')
                              OR (substr(:xdo_user_ui_locale, 1, 2) = 'zh' AND
substr(:xdo_user_ui_locale, 4, 2) IN ('CN', 'TW')))
            UNION SELECT locale_id FROM v_b_locale
            WHERE language = 'en'
                  AND NOT (substr(:xdo_user_ui_locale, 1, 2) in ('de', 'en',
'es', 'fr', 'it', 'ja', 'ko')
                        OR (substr(:xdo_user_ui_locale, 1, 2) = 'pt' AND
substr(:xdo_user_ui_locale, 4, 2) = 'BR')
                        OR (substr(:xdo_user_ui_locale, 1, 2) = 'zh' AND
```

```
substr(:xdo_user_ui_locale, 4, 2) IN ('CN', 'TW'))))
      ORDER BY t0.config_name) alias
      ON table.type_field = alias.num_value
…
```

In this code, `table.type_field` is the field containing a type code value that you want to replace with a string. Alias is the name you want to give the inner select clause. Finally, `enum_name` is the property name of the User Defined Enum.

To display in the report, you need to add *alias*.config_value to the select clause.

19.1.3 Adding Lists of Values

Add parameters to your report definition to enable your users to interact with the report and specify the data of interest from the data set.

To allow a user to select from a list of readable strings representing type codes, the report writer will need to create a List of Values (LOV) from a query on the User Defined Enums tables, filtered by the enum name.

19.1.3.1 User Defined Enums as List of Values for Filtering, English Only

The following listing shows how to write the query to populate the list of values.

```
SELECT enumElement.label, enumElement.num_value
FROM v_b_enum enum
      INNER JOIN v_b_enum_elmnt enumElement ON on enum.enum_id = enumElement.enum_
id
WHERE enum.prop_name = 'enum name'
ORDER BY enumElement.label
```

The following listing shows how to filter the report based on this LOV.

```
WHERE …
AND (:parameter IS NULL OR :parameter = table.type_field)
```

In these listings, `enum_name` is the property name of the User Defined Enum, `table.type_field` is the field containing a type code value that you want to replace with a string, and parameter is the named parameter. Review the *Oracle BI Publisher User's Guide* for information about creating and setting up report parameters.

19.1.3.2 User Defined Enums as List of Values for Filtering, Internalized

The following listing shows how to write the query to populate the list of values.

```
SELECT t0.config_value, element.num_value
FROM v_b_config_rb t0
INNER JOIN (
      SELECT enum_element.num_value, enum_element.str_value, enum.prop_name
      FROM v_b_enum enum
             INNER JOIN v_b_enum_elmnt enum_element ON enum.enum_id = enum_
element.enum_id
      WHERE enum.prop_name = 'enum name') element
      ON t0.config_name=element.prop_name || '.' || element.str_value || '.name'
WHERE t0.locale_id = (
      SELECT locale_id FROM v_b_locale
      WHERE language = substr(:xdo_user_ui_locale, 1, 2)
             AND country = substr(:xdo_user_ui_locale, 4, 2)
             AND (substr(:xdo_user_ui_locale, 1, 2) in ('de', 'en', 'es', 'fr',
'it', 'ja', 'ko')
                    OR (substr(:xdo_user_ui_locale, 1, 2) = 'pt' AND substr(:xdo_
```

```
user_ui_locale, 4, 2) = 'BR')
                  OR (substr(:xdo_user_ui_locale, 1, 2) = 'zh' AND substr(:xdo_
user_ui_locale, 4, 2) IN ('CN', 'TW')))
      UNION SELECT locale_id FROM v_b_locale
      WHERE language = substr(:xdo_user_ui_locale, 1, 2)
             AND NOT EXISTS(SELECT locale_id FROM v_b_locale
             WHERE language = substr(:xdo_user_ui_locale, 1, 2)
                   AND country = substr(:xdo_user_ui_locale, 4, 2))
                   AND country IS NULL
                   AND (substr(:xdo_user_ui_locale, 1, 2) in ('de', 'en', 'es',
'fr', 'it', 'ja', 'ko')
                         OR (substr(:xdo_user_ui_locale, 1, 2) = 'pt' AND
substr(:xdo_user_ui_locale, 4, 2) = 'BR')
                         OR (substr(:xdo_user_ui_locale, 1, 2) = 'zh' AND
substr(:xdo_user_ui_locale, 4, 2) IN ('CN', 'TW')))
      UNION SELECT locale_id FROM v_b_locale
      WHERE language = 'en'
             AND NOT (substr(:xdo_user_ui_locale, 1, 2) in ('de', 'en', 'es',
'fr', 'it', 'ja', 'ko')
                   OR (substr(:xdo_user_ui_locale, 1, 2) = 'pt' AND substr(:xdo_
user_ui_locale, 4, 2) = 'BR')
                   OR (substr(:xdo_user_ui_locale, 1, 2) = 'zh' AND substr(:xdo_
user_ui_locale, 4, 2) IN ('CN', 'TW'))))
ORDER BY t0.config_name
```

The filtering is performed in the same manner as the English Only version.

19.1.4 Adding Geolocation Data

The OAAM schema includes tables that map IP address ranges to location data including city, state, and country. The relevant tables are VCRYPT_IP_LOCATION_MAP, VCRYPT_CITY, VCRYPT_STATE, and VCRYPT_COUNTRY. Many tables contain IP addresses, and VCRYPT_IP_LOCATION_MAP contains foreign keys to each of VCRYPT_CITY, VCRYPT_STATE, and VCRYPT_COUNTRY.

In OAAM, IP addresses are stored as long numerals. The following listing shows how join a table containing an IP address to the VCRYPT_IP_LOCATION_MAP.

```
SELECT ...
FROM vcrypt_tracker_usernode_logs logs
      INNER JOIN vcrypt_ip_location_map loc ON (
             logs.remote_ip_addr >= loc.from_ip_addr AND logs.remote_ip_addr <=
loc.from_ip_addr
      )
```

For user input and display purposes, you will normally want to use the standard four-part IP address. The following listing shows how to display a numeric IP address as a standard IP, where *ipField* is the field or parameter containing the numeric IP address you want to display.

```
…
to_char(to_number(substr(to_char(ipField, 'XXXXXXXX'), 1, 3), 'XX')) || '.' ||
      to_char(to_number(substr(to_char(ipField, 'XXXXXXXX'), 4, 2), 'XX')) || '.'
||
      to_char(to_number(substr(to_char(ipField, 'XXXXXXXX'), 6, 2), 'XX')) || '.'
||
      to_char(to_number(substr(to_char(ipField, 'XXXXXXXX'), 8, 2), 'XX'))
...
```

The following listing shows how to convert a standard IP address to the long numeric format.

```
…
to_number(substr(ipField, 1, instr(ipField, '.')-1))*16777216 +
      to_number(substr(ipField, instr(ipField, '.', 1, 1)+1, instr(ipField, '.',
1, 2)-instr(ipField, '.', 1, 1)-1))*65536 +
      to_number(substr(ipField, instr(ipField, '.', 1, 2)+1, instr(ipField, '.',
1, 3)-instr(ipField, '.', 1, 2)-1))*256 +
      to_number(substr(ipField, instr(ipField, '.', 1, 3)+1))
```

19.1.5 Adding Sessions and Alerts

Sessions and alerts exist in the VCRYPT_TRACKER_USERNODE_LOGS and VCRYPT_ALERT tables, respectively. They join to each other via the REQUEST_ID field, and they each join to the geolocation data via the VCRYPT_IP_LOCATION_MAP table via the BASE_IP_ADDR field.

19.1.5.1 Type Code Lookups

The session table and the alert table have several type code fields that may be translated into readable text by following the instructions to look up the user defined enums by name. The following tables will list the type code fields and the name of the user defined enum.

Table 19–1 VCRYPT_TRACKER_USERNODE_LOGS

Field Name	User Defined Enum Name
AUTH_STATUS	auth.status.enum
AUTH_CLIENT_TYPE_CODE	auth.client.type.enum

Table 19–2 VCRYPT_ALERT

Field Name	User Defined Enum Name
ALERT_LEVEL	alert.level.enum
ALERT_TYPE	alert.type.enum
ALERT_STATUS	alert.status.enum
RUNTIME_TYPE	profile.type.enum

19.1.6 Example

This report will show a list of sessions, with user id, login id, auth status, and location. To start with, you will need to create two date parameters, fromDate and toDate. The query will look like the following:

```
SELECT s.request_id, s.user_id, s.user_login_id, auth.label, country.country_name,
state.state_name,
city.city_name
FROM vcrypt_tracker_usernode_logs s
      INNER JOIN vcrypt_ip_location_map loc ON s.base_ip_addr = loc.base_ip_addr
      INNER JOIN vcrypt_country country ON loc.country_id = country.country_id
      INNER JOIN vcrypt_state loc ON loc.state_id = country.state_id
      INNER JOIN vcrypt_city city ON loc.city_id = city.city_id
      LEFT OUTER JOIN (
            SELECT enumElement.num_value, enumElement.label
```

```
            FROM v_b_enum enum
                  INNER JOIN v_b_enum_elmnt enumElement ON on enum.enum_id =
enum_element.enum_id
            WHERE enum.prop_name = 'auth.status.enum') auth
            ON s.auth_status = auth.num_value
WHERE (:fromDate IS NULL OR s.create_time >= :fromDate)
      AND (:toDate IS NULL OR s.create_time <= :toDate)
ORDER BY s.create_time DESC
```

19.1.7 Adding Layouts to the Report Definition

BI Publisher offers several options for designing templates for your reports. Refer to the *Oracle Business Intelligence Publisher Report Designer's Guide* for instructions at the following URL:

http://download.oracle.com/docs/cd/E12844_01/doc/bip.1013/e12187/toc.htm

19.2 Building OAAM Transactions Reports

This section explains how you can build transaction reports. It contains the following topics:

- Get Entities and Transactions Information
- Discover Entity Data Mapping Information
- Discover Transaction Data Mapping Information
- Build Reports

19.2.1 Get Entities and Transactions Information

To get the Transaction Definition key and Entity Definition keys, follow these steps:

1. Log into OAAM Admin and go to the Transactions menu and search for the transaction definitions you are interested in.
2. Go to the **General** tab and note down the **Definition Key** of the transaction. This is the "Transaction Definition Key" of the transaction.
3. Go to the **Entities** tab of the transaction and note down the distinct list **Entity Name**.
4. Choose the **Entities** menu option to search for Entities and note the **Key** of each of those entities. That is the "Entity Definition Key" of the entities.

19.2.2 Discover Entity Data Mapping Information

To discover entity data mapping information that you will need to create your report, follow the procedures in this section.

19.2.2.1 Information about Data Types

For your reference, number data types are listed in the following table.

Table 19–3 Information about Data Types

Data Type	Description
1	Represents String data
2	Represents Numeric data. Data stored is equal to (Original value * 1000).
3	Date type data. Store the data in "'YYYY-MM-DD HH24:MI:SS TZH:TZM" format and also retrieve it using same format.
4	Boolean data. Stored as strings. "True" represents TRUE and "False" represents FALSE

19.2.2.2 Discover Entity Data Details Like Data Type, Row and Column Mappings

To get the entity data details that you will need to construct your report, follow these steps:

1. Get the Entity Definition Key by looking at the entity definition using the OAAM Admin Console.
2. Get details of how entity data is mapped using the SQL Query:

```
SELECT label,
  data_row,
  data_col,
  data_type
FROM vt_data_def_elem
WHERE status =1
AND data_def_id =
  (SELECT data_def_id
  FROM vt_data_def_map
  WHERE relation_type   ='data'
  AND parent_obj_type   =3
  AND parent_object_id IN
    (SELECT entity_def_id
    FROM vt_entity_def
    WHERE entity_def_key=<Entity Definition Key>
    AND status =1
    )
  )
ORDER BY data_row ASC,
  data_col ASC;
```

19.2.2.3 Build Entity Data SQL Queries and Views

The above SQL query gives a list of data fields of the entity with data type and row, column position. Using that information, build a SQL query based on the following information that represents data of the given entity. It is also recommended to create/build a view based on this SQL query that represents data of the given entity.

> **Note:** EntityRowN represents an entity data row. If your entity has 3 distinct `data_row` values from the above query then you would have 3 EntityRows, name the aliases as `EntityRow1`, `EntityRow2`, and so on, and similarly take care of the corresponding joins as shown below.

```
SELECT ent.ENTITY_ID,
    ent.EXT_ENTITY_ID,
    ent.ENTITYNAME,
    ent.ENTITY_KEY,
    ent.ENTITY_TYPE,
```

```
    EntityRowN<row>.DATA<col> <column_name>,
    (EntityRowN<row>.NUM_DATA<col>/ 1000.0) <numeric_column_name>,
    to_timestamp_tz(EntityRowN<row>.DATA<col>, 'YYYY-MM-DD HH24:MI:SS TZH:TZM')
<date_column_name>,
    ent.CREATE_TIME,
    ent.UPDATE_TIME,
    ent.EXPIRY_TIME,
    ent.RENEW_TIME
  FROM
    VT_ENTITY_DEF entDef,
    VT_ENTITY_ONE ent
    LEFT OUTER JOIN VT_ENTITY_ONE_PROFILE EntityRowN
  ON (EntityRowN.ENTITY_ID    = ent.ENTITY_ID
  AND EntityRowN.ROW_ORDER    = <row>
  AND EntityRowN.EXPIRE_TIME IS NULL)
    LEFT OUTER JOIN VT_ENTITY_ONE_PROFILE EntityRowN+1
  ON (EntityRowN+1.ENTITY_ID    = ent.ENTITY_ID
  AND EntityRowN+1.ROW_ORDER    = <row+1>
AND row1.EXPIRE_TIME IS NULL)
  WHERE
ent.ENTITY_DEF_ID      = entDef.ENTITY_DEF_ID and
entDef.ENTITY_DEF_KEY=<Entity Definition Key>
```

19.2.3 Discover Transaction Data Mapping Information

To discover transaction data mapping information that you will need to create your report, follow the procedures in this section.

19.2.3.1 Discover Transaction data details like Data Type, Row and Column mappings

To get entity data details you will need to construct your report, follow these steps:

1. Get list of transaction to entity definition mapping Ids using the following SQL:

```
SELECT map_id
FROM
vt_trx_ent_defs_map,
  vt_trx_def
WHERE
vt_trx_ent_defs_map.trx_def_id = vt_trx_def.trx_def_id
AND vt_trx_def.trx_def_key  =<Transaction Definition Key>
```

2. Use the following SQL query to get details of all transaction data fields, their data type and their row, column mapping:

```
SELECT label,
  data_row,
  data_col,
  data_type
FROM vt_data_def_elem
WHERE status    =1
AND data_def_id =
  (SELECT data_def_id
  FROM vt_data_def_map
  WHERE relation_type   ='data'
  AND parent_obj_type   =1
  AND parent_object_id IN
    (SELECT trx_def_id
    FROM vt_trx_def
    WHERE trx_def_key='mayo_pat_rec_acc'
```

```
        AND status       =1
        )
      )
    ORDER BY data_row ASC,
      data_col ASC;
```

19.2.3.2 Build Transaction Data SQL Queries and Views

Use the information from the previous section and build a SQL query that represents transaction data based on the following:

Note: It is recommended to build a view based on this Query so that it is easier to build reports

```
SELECT trx.LOG_ID,
    trx.USER_ID,
    trx.REQUEST_ID,
    trx.EXT_TRX_ID,
    trx.TRX_TYPE,
    trx.STATUS,
    trx.SCORE,
    trx.RULE_ACTION,
    trx.TRX_FLAG,
    trx.POST_PROCESS_STATUS,
    trx.POST_PROCESS_RESULT,
    TxnDataRowN<row>.DATA<col> <data_column_name>,
   (TxnDataRowN<row>.NUM_DATA<col>/ 1000.0) <numeric_column_name>,
    to_timestamp_tz(TxnDataRowN<row>.DATA<col>, 'YYYY-MM-DD HH24:MI:SS TZH:TZM')
<date_column_name>,
    (SELECT entTrxMap.MAP_OBJ_ID
    FROM VT_ENT_TRX_MAP entTrxMap
    WHERE entTrxMap.DEF_MAP_ID = <Transaction to Entity Mapping Id of Entity1_
Name>
    AND entTrxMap.TRX_ID       = trx.LOG_ID
    ) <EntityN_Name>,
    (SELECT entTrxMap.MAP_OBJ_ID
    FROM VT_ENT_TRX_MAP entTrxMap
    WHERE entTrxMap.DEF_MAP_ID = <Transaction to Entity Mapping Id of Entity2_
Name>
    AND entTrxMap.TRX_ID       = trx.LOG_ID
    ) <EntityN+1_Name>,
    trx.CREATE_TIME,
    trx.UPDATE_TIME,
    TRUNC(trx.create_time, 'HH24') created_hour,
    TRUNC(trx.create_time, 'DDD') created_day,
    TRUNC(trx.create_time, 'DAY') created_week,
    TRUNC(trx.create_time, 'MM') created_month,
    TRUNC(trx.create_time, 'YYYY') created_year
  FROM VT_TRX_DEF trxDef,
    VT_TRX_LOGS trx
  LEFT OUTER JOIN VT_TRX_DATA TransactionDataRowN
  ON (TransactionDataRowN.TRX_ID         = trx.LOG_ID
  AND TransactionDataRowN.ROW_ORDER      = <rowN>)
LEFT OUTER JOIN VT_TRX_DATA TransactionDataRowN+1
  ON (TransactionDataRowN+1.TRX_ID         = trx.LOG_ID
  AND TransactionDataRowN+1.ROW_ORDER      = <rowN+1>)
  WHERE trx.TRX_DEF_ID      = trxDef.TRX_DEF_ID  and
trxDef.TRX_DEF_KEY=<Transaction Definition Key>
```

19.2.4 Build Reports

Follow the instructions in this section to build reports for entities and transactions.

19.2.4.1 Building Entity Data Reports

Use the SQL Queries or Views built using the information mentioned in Section 19.2.2.3, "Build Entity Data SQL Queries and Views."

19.2.4.2 Building Transaction Data Reports

Use the SQL Queries or Views built using the information mentioned in Section 19.2.3.2, "Build Transaction Data SQL Queries and Views."

19.2.4.3 Joining Entity Data Tables and Transaction data tables

You can join the transaction data views you built with entity data view using `VT_ENT_TRX_ MAP.MAP_OBJ_ID` which is indicated using the pseudo column `<EntityN_Name>`.

20

Developing Custom Challenge Processors

The OAAM Server provides a challenge processor framework that allows for custom implementations of challenge mechanisms.

This chapter contains the following sections:

20.1 What are Challenge Processors

A challenge processor is java code that implements the `ChallengeProcessorIntf` interface or extends the `AbstractChallengeProcessor` class.

Challenge processors can be created to perform the following tasks for a challenge:

- Generate challenge secret (password) to send to the user.
- Validate the user answer
- Control delivery wait page (if needed)
- Check if delivery service is available (if needed)

For example, 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 by a challenge type.

20.2 Code Challenge Processors

This section contains information on the challenge processor class and methods to implement. An implementation example is also provided for your reference.

20.2.1 Class

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.

For instructions on customizing, extending, or overriding Oracle Adaptive Access Manager properties, refer to Chapter 7, "Customizing Oracle Adaptive Access Manager."

20.2.2 Methods

The methods used in a challenge processor are listed in the sections following.

Table 20–1 Challenge Processor Methods

Methods	Description
protected boolean generateSecret(UIOSessionData sessionData, boolean isRetry)	This method is used to generate code to send to client
protected boolean validateAnswer(UIOSessionData sessionData, String answer)	This method is used to validate the user answer.
public String checkDeliveryStatus(UIOSessionData sessionData, boolean userWaiting, boolean isRetry)	This method is used if you want to provide a wait until message is sent.
public boolean isServiceAvailable(UIOSessionData sessionData)	This method is used to check if external service is available.

20.2.3 Example: Email Challenge Processor Implementation

An implementation of the email 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 isRety) {
   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;
  }
}
```

20.2.4 Secret (PIN) Implementation

The `AbstractOTPChallengeProcessor` class has a default pin generation method, `generateCode`, that you can override to provide your pin generation logic.

20.3 Define the Delivery Channel Types for the Challenge Processors

This section contains instructions on defining a delivery channel type. Examples are provided for your reference.

20.3.1 Challenge Type Enum

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>`

For example,

`bharosa.uio.default.challenge.type.enum.ChallengeEmail`

The challenge type enum is used to associate a challenge type with the java code needed to perform any work related to that challenge type. An example of

implementing an email challenge processor is shown in Section 20.2.3, "Example: Email Challenge Processor Implementation."

The Challenge Type ID (for example, ChallengeEmail) should match a rule action returned by the rules when that challenge type is used. The rule action for ChallengeEmail is rule.action.enum.ChallengeEmail. The rule action is to challenge the user using email using the email delivery channel. "Channel" normally refers to the delivery channel used to send to the user.

20.3.2 Example: Defining an OTP Channel Type

To define a challenge type, use the following property:

```
bharosa.uio.default.challenge.type.enum.MyChallenge
```

In the property, **default** is the UIO application name, and **MyChallenge** is the Challenge Type being added. For example, **ChallengeEmail** is the Challenge Type in the example below.

```
bharosa.uio.default.challenge.type.enum.ChallengeEmail
```

The rule action is to challenge the user with email using the email delivery channel.

```
rule.action.enum.ChallengeEmail
```

To enable/disable a challenge type, the available flag should be set:

```
bharosa.uio.default.challenge.type.enum.MyChallenge.available = false
```

Table 20–2 Challenge type Flags

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

Setting the **available** flag and setting the **enabled** flag are different. The **enabled** flag would remove it from list.

Example for Defining a Channel Type

Attributes bharosa.uio.default.challenge.type.enum with example values are 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
```

20.4 Configure User Input Properties

Instructions to configure user information properties are in the following sections:

- Enable Registration and Preferences Input
- Set Contact Information Inputs

For instructions on customizing, extending, or overriding Oracle Adaptive Access Manager properties, refer to Chapter 7, "Customizing Oracle Adaptive Access Manager."

20.4.1 Enable Registration and Preferences Input

Default configurations for enabling for registration and preference input are listed as follows:

Contact information registration

```
bharosa.uio.default.register.userinfo.enabled=false
```

Contact information preferences

```
bharosa.uio.default.userpreferences.userinfo.enabled=false
```

20.4.2 Set Contact Information Inputs

If user information registration and user preferences are true, configure input information.

Contact information inputs are defined in userinfo.inputs.enum. The enum element is:

```
bharosa.uio.<application>.userinfo.inputs.enum.<inputname>
```

Table 20–3 Properties for Contact Input

Property	Description
inputname	Name used for the input field in the HTML form
inputtype	Set for text or password input
maxlength	Maximum length of user input
required	Set if the field is required on the registration page
order	The order displayed in the user interface
regex	Regular expression used to validate user input for this field
errorCode	Error code used to look up validation error message (bharosa.uio.<application ID>.error.<errorCode>)
managerClass	java class that implements com.bharosa.uio.manager.user.UserDataManagerIntf (if data is to be stored in Oracle Adaptive Access Manager database this property should be set to com.bharosa.uio.manager.user.DefaultContactInfoManager)

Email Input 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
```

20.5 Configure the Challenge Pads Used for Challenge Types

By default, challenge devices that will be used are configured through rules. The rules are under the AuthentiPad checkpoint where you can specify the type of device to use based on the purpose of the device.

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.

Table 20–4 Authentication Device Type

Property	Description
None	No HTML page or authentication pad
DeviceKeyPadFull	Challenge user using KeyPad.
DeviceKeyPadAlpha	Challenge user with the alphanumeric KeyPad (numbers and letters only, no special characters)
DeviceTextPad	Challenge user using TextPad.
DeviceQuestionPad	Challenge user using QuestionPad.
DevicePinPad	Challenge user using PinPad.
DeviceHTMLControl	Challenge user using HTML page instead of an authentication pad.

21

Creating a View of a Non-OAAM Database

Users who want to load from a non-OAAM database will need to create a view in their remote data source. This document explains how to create this view.

21.1 The OAAM_LOAD_DATA_VIEW

The Out-of-the-Box Loader for OAAM Offline requires a table or view with a specific name and structure to exist in the load data source. The structure is given in the following table.

Table 21–1 OAAM_LOAD_DATA_VIEW

Field Name	Data Type	Description
LOGIN_TIMESTAMP	Date/Time	The login time.
SESSION_ID	Character	Uniquely identifies a login record.
USER_ID	Character	The user's User ID.
LOGIN_ID	Character	The user's Login ID. This may be the same as the USER_ID if the load datasource does not distinguish between User ID and Login ID.
DEVICE_ID	Character	Identifies the user's device.
GROUP_ID	Character	The user's primary user group, or an application ID.
IP_ADDRESS	Integer	The IP address, in the form of a long integer.
AUTH_STATUS	Integer	The auth status. If loading from a non-OAAM schema, this field should be a decode function that converts the remote data source's authentication status into an OAAM authentication status, defined by the user defined enum auth.status.enum. If the remote schema has no concept of auth status, then this value should be -1.
CLIENT_TYPE	Integer	The client type. When loading from a non-OAAM schema, this should be -1.
USER_AGENT	Character	The user agent string from the browser.
FLASH_FINGERPRINT	Character	This field represents the digital fingerprint. It may be null if not supported by the load datasource.
DIGITAL_COOKIE	Character	This field represents the digital cookie set by OAAM. When loading from a non-OAAM schema, this should be null.

Table 21–1 (Cont.) OAAM_LOAD_DATA_VIEW

Field Name	Data Type	Description
EXP_DIGITAL_COOKIE	Character	This field represents the expected digital cookie set by OAAM. When loading from a non-OAAM schema, this should be null.
SECURE_COOKIE	Character	This field represents the secure cookie set by OAAM. When loading from a non-OAAM schema, this should be null.
EXP_SECURE_COOKIE	Character	This field represents the expected secure cookie set by OAAM. When loading from a non-OAAM schema, this should be null.

21.2 Schema Examples

The OAAM Schema and custom schema are shown below.

21.2.1 OAAM Schema

The following example shows the SQL for the `OAAM_LOAD_DATA_VIEW` that ships with OAAM.

```
CREATE OR REPLACE FORCE VIEW OAAM_LOAD_DATA_VIEW (
LOGIN_TIMESTAMP, SESSION_ID, USER_ID, LOGIN_ID, DEVICE_ID, GROUP_ID,
      IP_ADDRESS, AUTH_STATUS, CLIENT_TYPE, USER_AGENT, FLASH_FINGERPRINT,
      DIGITAL_COOKIE, EXP_DIGITAL_COOKIE, SECURE_COOKIE, EXP_SECURE_COOKIE) AS
SELECT l.create_time LOGIN_TIMESTAMP, l.request_id SESSION_ID, l.user_id USER_ID,
   l.user_login_id LOGIN_ID, l.node_id DEVICE_ID, l.user_group_id GROUP_ID,
   l.remote_ip_addr IP_ADDRESS, l.auth_status AUTH_STATUS, l.auth_client_type_code
      CLIENT_TYPE,
   (SELECT t1.data_value FROM v_fprints t1 WHERE t1.fprint_id=l.fprint_id) USER_
AGENT,
   (SELECT t2.data_value FROM v_fprints t2 WHERE t2.fprint_id=l.digital_fp_id)
      FLASH_FINGERPRINT,
   l.sent_dig_sig_cookie DIGITAL_COOKIE, l.expected_dig_sig_cookie EXP_DIGITAL_
COOKIE,
   l.sent_secure_cookie SECURE_COOKIE, l.expected_secure_cookie EXP_SECURE_COOKIE
FROM vcrypt_tracker_usernode_logs l;
```

For discussion purposes, consider this statement in two parts.

The first part starts at the beginning and ends before the Select. This part is required and cannot be modified.

The second part starts with the Select and continues to the end of the statement. If loading from a non-OAAM schema, this part would be customized to select data from that schema.

21.2.2 Custom Schema Example

In this example, you would want to load from a table that looks like the following. You would want to have "Banking" as your primary group or Application ID, and you would not want to load test data.

LOGINS

Table 21–2 LOGINS

Field Name	Data Type	Description
LOGIN_TIME	Date/Time	The login time.
LOGIN_ID	Integer	Primary Key
USER_NAME	Character	The user's Login ID.
DEVICE_ID	Character	Identifies the user's device.
IP_ADDRESS	Character	The IP address, in dot notation.
AUTH_STATUS	Character	'S' = Success, 'I' = Invalid User, 'F' = Wrong Password.
USER_AGENT	Character	The user agent string from the browser.
IS_TEST	Integer	0 = Real Data, 1 = Test data

In this case, a decode statement is needed to convert the custom authentication status to an OAAM authentication status, and the IP address needs to be parsed to convert it into a long integer. A view must be created that looks like the following.

```
CREATE OR REPLACE FORCE VIEW OAAM_LOAD_DATA_VIEW (
LOGIN_TIMESTAMP, SESSION_ID, USER_ID, LOGIN_ID, DEVICE_ID, GROUP_ID,
      IP_ADDRESS, AUTH_STATUS, CLIENT_TYPE, USER_AGENT, FLASH_FINGERPRINT,
      DIGITAL_COOKIE, EXP_DIGITAL_COOKIE, SECURE_COOKIE, EXP_SECURE_COOKIE) AS
SELECT l.login_time LOGIN_TIMESTAMP, cast(l.login_id AS varchar2(256)) SESSION_ID,
   l.user_name USER_ID, l.user_name, LOGIN_ID, l.device_id DEVICE_ID,
   'Banking' GROUP_ID,
   to_number(substr(l.ip_address, 1, instr(l.ip_address, '.')-1))*16777216
      to_number(substr(l.ip_address, instr(l.ip_address, '.', 1, 1)+1,
          instr(l.ip_address, '.', 1, 2)-instr(l.ip_address, '.', 1, 1)-1))*65536
      to_number(substr(l.ip_address, instr(l.ip_address, '.', 1, 2)+1,
          instr(l.ip_address, '.', 1, 3)-instr(l.ip_address, '.', 1, 2)-1))*256
      to_number(substr(l.ip_address, instr(l.ip_address, '.', 1, 3)+1)) IP_
ADDRESS,
   decode(l.auth_status, 'S', 0,
      'I', 1,
      'F', 2,
          -1) AUTH_STATUS,
   -1 CLIENT_TYPE, l.user_agent USER_AGENT, null FLASH_FINGERPRINT,
   null DIGITAL_COOKIE, null EXP_DIGITAL_COOKIE, null SECURE_COOKIE,
   null EXP_SECURE_COOKIE
FROM logins l
WHERE l.is_test = 0
```

Here, you map your `user_name` to `USER_ID` and `LOGIN_ID`, you map a literal string "Banking" to `GROUP_ID`, you parse your `ip_address` string and convert it to a long integer, you use a decode statement to convert your `auth_status`, you map `-1` to `CLIENT_TYPE`, and you map literal `null` to `FLASH_FINGERPRINT`, `DIGITAL_COOKIE`, `EXP_DIGITAL_COOKIE`, `SECURE_COOKIE`, and `EXP_SECURE_COOKIE`.

22

Developing a Custom Loader for OAAM Offline

This chapter describes the overall data loader framework for OAAM Offline:

- Basic framework and the default implementation
- How to override the default functionality

This document assumes that you are familiar with the concepts of OAAM Offline.

22.1 Base Framework

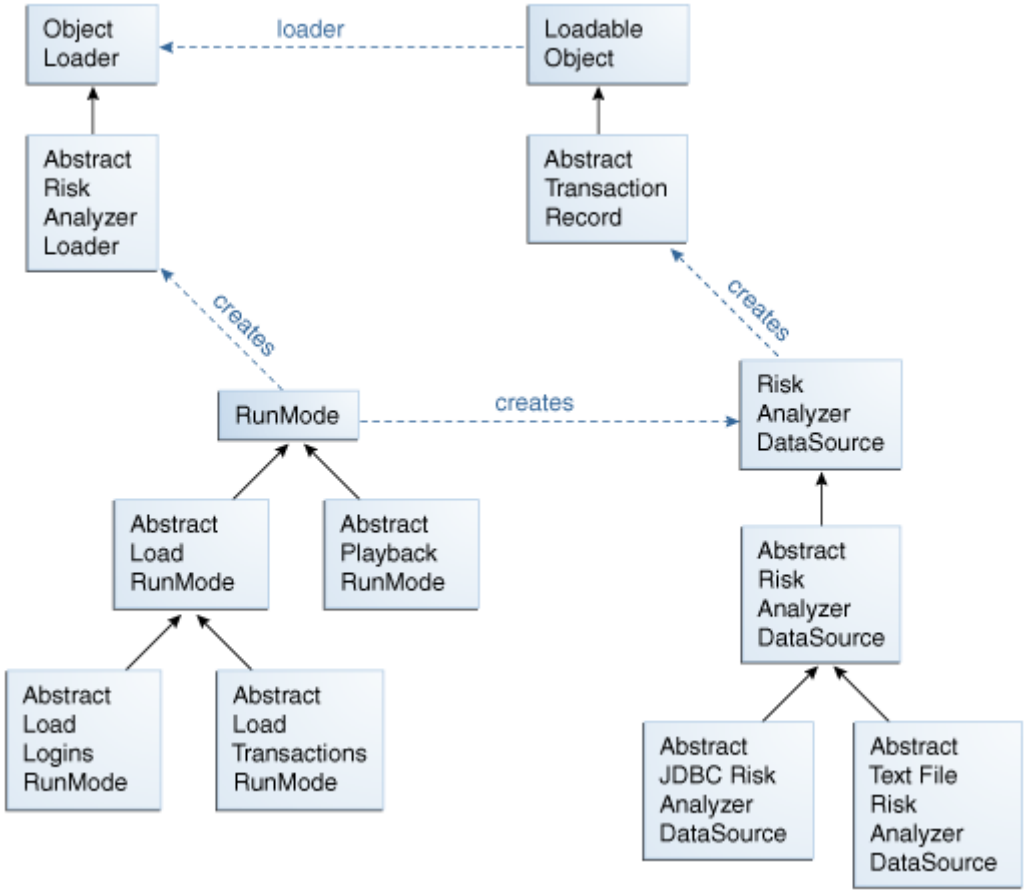
A custom loader is required only if the data from sources other than a database, data other than login, or complex data is needed for the OAAM Offline task.

22.1.1 Overview

The OAAM Offline custom loader consists of the following key parts:

- loadable object
- data source
- loader
- run modes

Figure 22–1 Basic Framework of a Custom Loader



The loadable object represents an individual data record. The data source represents the entire store of data records and the loader processes the records. There are two types of run mode: load and playback. The run modes encapsulate the differences between loading a Session Set and running a Session Set.

22.1.2 Important Classes

Table 22–1 provides a summary of the different data loader classes.

Table 22–1 Data Loader Classes

Class	Description
RunMode	There are two basic types of RunMode: load and playback. Load run modes are responsible for importing session set data into the OAAM Offline system, and the playback run mode is responsible for processing preloaded session set data. Each run mode is responsible for constructing data source and loader. An additional responsibility is determining how to start where a previous job ended, in the cases of recurring schedules of autoincrementing session sets or paused and resumed run sessions. `AbstractLoadRunMode` and `AbstractPlaybackRunMode` each have a factory method named `getInstance()`. These methods check to see if the default run modes have been overridden.
RiskAnalyzerDataSource	The `RiskAnalyzerDataSource` is responsible for acquiring the data and iterating through it. `RiskAnalyzerDataSource` has two abstract implementors: `AbstractJDBCRiskAnalyzerDataSource` and `AbstractTextFile-RiskAnalyzerDataSource`. The `AbstractJDBCRiskAnalyzerDataSource` implements the base functionality for iterating through a JDBC result set, and the `AbstractTextFileRiskAnalyzerDataSource` implements the base functionality for iterating through a text file.
AbstractTransactionRecord	The `AbstractTransactionRecord` class only contains the state and behavior required to manage the overall risk analysis process. Subclasses will add additional state and behavior to satisfy client requirements.
AbstractRiskAnalyzerLoader	The `AbstractRiskAnalyzer`Loader is the base implementation of ObjectLoader for the Risk Analyzer process. It provides basic exception handling, but otherwise leaves the implementation up to its subclasses.

22.1.3 General Framework Execution

The following pseudocode shows the general framework execution.

```
AbstractRiskAnalyzerLoader loader = runMode.buildObjectLoader();
RiskAnalyzerDataSource dataSource = runMode.acquireDataSource();
try{
   while (dataSource.hasMoreRecords()) {
      AbstractTransactionRecord eachRecord = dataSource.nextRecord();
      loader.process(eachRecord);
   }
} finally {
   dataSource.close();
}
```

22.2 Default Implementation

The default implementation for the Risk Analyzer data loader framework works as follows:

Load mode: When in load mode, it uses any database as a data source, it expects login data, and it performs device fingerprinting.

Playback mode: When in playback mode, it uses the `VCRYPT_TRACKER_USERNODE_LOGS` and `V_FPRINTS` tables as its data source, and it runs each record through all active models.

22.2.1 Default Load Implementation

The default load implementation is summarized below.

Figure 22–2 Default Load Implementation

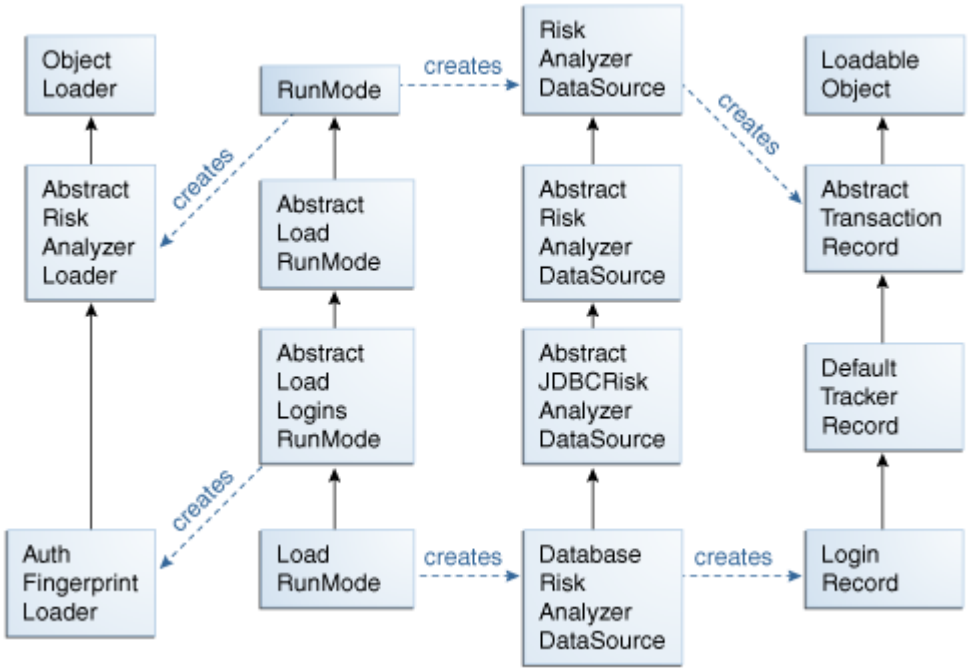


Table 22–2 Default Implementation

Components	Description
LoadRunMode	The default `LoadRunMode` class instantiates a `DatabaseRiskAnalyzerDatasource` as its data source and a `AuthFingerprintLoader` as its loader.
DatabaseRiskAnalyzerDatasource	The `DatabaseRiskAnalyzerDatasource` creates `LoginRecords` from a JDBC data source. It uses a set of configuration properties to tell it how to connect to the JDBC data source and to tell it how to build a `LoginRecord` from the tables and fields in the remote database. The default values for these properties map to the tables in an OAAM database.
LoginRecord	The login record contains all of the available fields required to call the methods for device fingerprinting on the TrackerAPIUtil class.
AuthFingerprintLoader	The `AuthFingerprintLoader` uses the data in the `LoginRecord` to simulate a login. This causes the system to perform device fingerprinting, run device identification time rules, and store the user node log and fingerprint data in the OAAM Offline database.

22.2.2 Default Playback Implementation

The default playback implementation is summarized below.

Figure 22–3 Default Playback Implementation

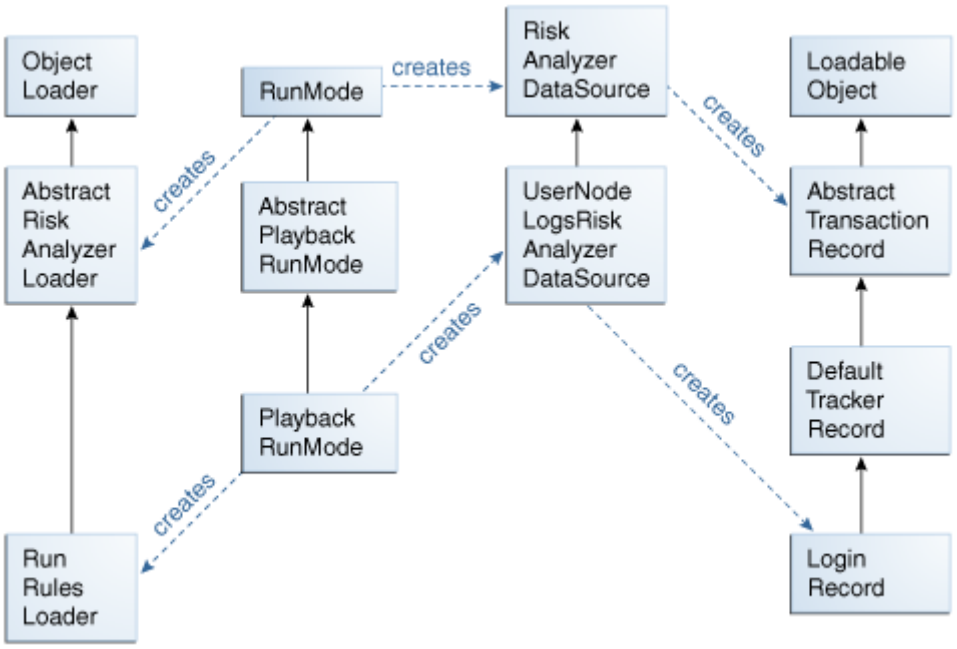


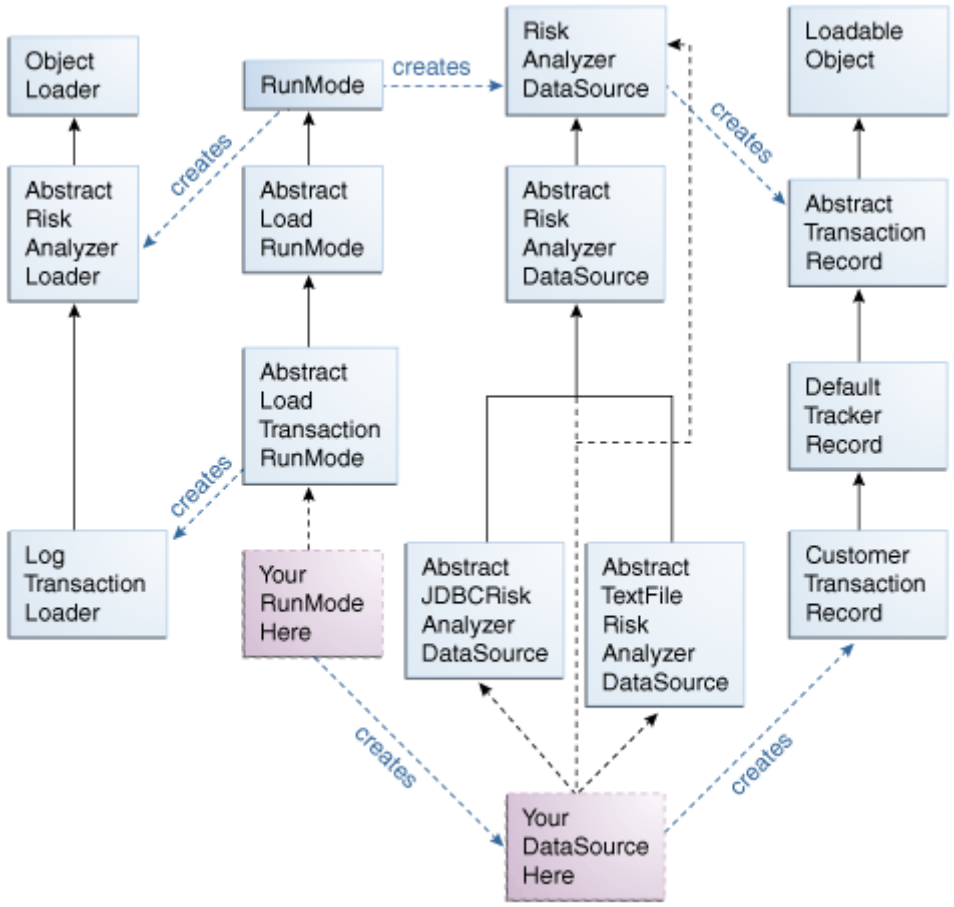
Table 22–3 Default Playback Implementation

Components	Description
PlaybackRunMode	The default `PlaybackRunMode` class instantiates a `UserNodeLogsRiskAnalyzerDataSource` as its data source and a `RunRulesLoader` as its loader.
UserNodeLogsRiskAnalyzerDatasource	The `UserNodeLogsRiskAnalyzerDatasource` creates `LoginRecords` from the `VCRYPT_TRACKER_USERNODE_LOGS` and `V_FPRINTS` tables in the OAAM Offline database.
LoginRecord	The login record contains all of the fields required to call the methods for rules processing on the `TrackerAPIUtil` class.
RunRulesLoader	The `RunRulesLoader` processes pre-auth rules on all `LoginRecords`, and processes post-auth rules on all `LoginRecords` with a successful authentication status.

22.3 Implementation Details: Overriding the Loader or Playback Behavior

There are several cases that would require the default behavior to be overridden. You would need to override the default loading behavior to load data from a source other than a database or to load transactional data into the system. You would need to override the default playback behavior if you needed to perform a procedure other than rules processing.

Figure 22–4 Overriding the Loader or Playback Behavior



22.4 Implement RiskAnalyzerDataSource

If you are loading login data from a data source other than a JDBC database, or if you are loading transactional data, then you will need to create your own subclass of `RiskAnalyzerDataSource`. There are three ways to do this: extending `AbstractJDBCRiskAnalyzerDataSource`, extending `AbstractTextFile-RiskAnalyzerDataSource`, or extending `AbstractRiskAnalyzerDataSource`.

22.4.1 Extending AbstractJDBCRiskAnalyzerDataSource

This is the appropriate choice if you are loading any sort of data through a JDBC connection. It includes default behavior for opening a JCBC connection, issuing a subclass specified SQL query to build a JDBC result set, and querying the database for a count of the total number of records.

There are three abstract methods that you have to implement.

- buildBaseSelect() returns the SQL query you will use to read the data. It should not include any order by statement. The superclass will use your implementation of `getOrderByField()` to add the order by statement.

- `getOrderByField()` returns the name of the database field that your query should be sorted on. This is usually the date field.
- `buildNextRecord()` turns one or more records from the JDBC result set into your loadable data record.

There are protected fields in the superclass available for your use, and you will need them when you implement the abstract methods. The most important is `resultSet`, which refers to your JDBC result set. When `hasMoreRecords()` has been called and returns true, you are guaranteed that `resultSet` is in a valid state and pointing at the current record. In addition, when you implement `buildNextRecord()`, you can safely assume that `resultSet` is in a valid state and pointing at the current record.

Other fields you might need to know about are connection and controller. connection refers to your JDBC to the remote database. controller is an instance of RiskAnalyzer and contains context information about your current OAAM Offline job.

Other methods that you can override if the default behavior is not what you need are `buildConnection()`, `buildSelectCountStatement()`, `getTotalNumberToProcess()`, and `buildSelectStatement()`.

You would `override buildConnection()` if you wanted to change how you instantiate the remote JDBC connection.

You would override `buildSelectCountStatement()` if you wanted to change the SQL used to count the number of records to be read in.

You would override `getTotalNumberToProcess()` if you wanted to replace the algorithm that returns the number of records to be read in. You would only do this if overriding `buildSelectCountStatement()` was not enough to give you the behavior you need.

Finally, you would override `buildSelectStatement()` if you wanted to make changes to the SQL used to read the records from the remote databases, such as changing how the order by clause is applied.

22.4.2 Extending AbstractTextFileAnalyzerDataSource

This is the appropriate choice if you are loading data from a text file. It includes default behavior for opening a file, skipping to the first line in the file whose date is at or above the beginning of the session set's range, and stopping when you reach a line whose date is above the end of the session set's range. For this scheme to work, the input file must be sorted by date. There are two abstract methods you will have to implement.

`dateOfCurrentRecord()` tells us the date of the current record. When this method is called, you are guaranteed that the `currentLine` field is valid and contains the current line of your file.

`buildNextRecord()` turns one or more lines from the file into your loadable data record.

If your input file contains lines that you need to ignore, you can override the `lineIsBad()` method to examine the text passed to it, and return true if the line is one that needs to be skipped.

There are protected fields in the superclass available for your use, and you will need them when you implement the abstract methods. The two most important ones are `currentLine`, which refers the last read in line from your file, and reader, which is a `java.io.BufferedReader` representing a character stream your file and is pointed at the next line after `currentLine`, if any. When `hasMoreRecords()` has been

called and returns true, you are guaranteed that `currentLine` is not null and is within your session set's range.

Another field you might need to know about is controller. controller is an instance of `RiskAnalyzer` and contains context information about your current OAAM Offline job.

Other methods that you can override if the default behavior is not what you need are `getTotalNumberToProcess()`, `isBeforeBeginning()`, `isAfterEnd()`, and `skipToBeginning()`.

The default implementation of `getTotalNumberToProcess()` assumes that you will not know how many records you intend to process, so it returns a constant invalid value, signifying N/A. If your implementation has some way to know how many records it will be processing, you can override this method with your algorithm.

The default implementation of `isBeforeBeginning()` returns true if the date returned by `dateOfCurrentRecord()` is before controller's start date range. You can override this method if you need to do this in a different way. Your file must be sorted by whichever field or fields you are comparing here.

The default implementation of `isAfterEnd()` returns true if the date returned by `dateOfCurrentRecord()` is after controller's end date range. You can override this method if you need to do this in a different way. Your file must be sorted by whichever field or fields you are comparing here. If you wanted to turn off this behavior, you can override this method to return a constant value of false.

The default implementation of `skipToBeginning()` iterates through the file until it finds a record for which `isBeforeBeginning()` returns false. If you wanted to turn off this behavior, you could override this method to have an empty body. The default manner in which this method skips to the next record is to go through the motions of constructing the loadable record, only to throw it away. This step is required for the case where it takes multiple lines from the file to make a single loadable record, to ensure that all of those lines are skipped at once. If you are guaranteed that one line is equal to one loadable record, you can replace this method with the following more efficient version. The `nextRecordIsReady` variable is a flag the `hasMoreRecords()` checks to know if it needs to skip to the next line. Any overriding version of `skiptoBeginning()` must set the initialized flag to true before calling `hasMoreRecords()` to avoid an infinite loop.

```
protected void skipToBeginning() throws Exception {
initialized = true;
   while (hasMoreRecords() && isBeforeBeginning()) {
      nextRecordIsReady = false;
   }
}
```

22.4.3 Extending AbstractRiskAnalyzerDataSource

If neither `AbstractJDBCRiskAnalyzerDataSource` nor `AbstractTextFile-RiskAnalyzerDataSource` is appropriate, then you will need to extend `AbstractRiskAnalyzerDataSource` instead. You might find yourself in this situation if you are reading from a binary file or if you are implementing a data source for a custom playback mode and using TopLink to read from the OAAM Offline database.

The constructor should put your class into a state so that you are ready to iterate through the data. There are four abstract methods you will have to implement.

`getTotalNumberToProcess()` will return the total number of records in the data source that satisfy the conditions that define a given Session Set.

`hasMoreRecords()` will return true if there are more records to be processed, and will move any sort of record pointer to the next available record if required. There is a flag named `nextRecordIsReady` that should be used for signaling here. The superclass sets this flag to false when it has made use of the next available record. Your implementation of `hasMoreRecords()` should check the value of the `nextRecordIsReady` flag, move the pointer to the next record only if the flag's value is false, and change the flag's value to true when you successfully move the pointer to a new record. If you are following this paradigm, then if your implementation of `hasMoreRecords()` is called while `nextRecordIsReady` is true, then you should return true without changing the state of any record pointers.

`buildNextRecord()` will return a new instance of the required subclass of `AbstractTransactionRecord`.

`close()` is called when you have finished processing all of the records. Any required clean-up should be performed here.

22.5 Implement RunMode

If you have created any customized classes for the load or playback behavior, you are required to create a customized subclass of `AbstractLoadLoginsRunMode`, `AbstractLoadTransactionsRunMode`, or `PlaybackRunMode`, depending on your requirements.

The most important `RunMode` methods are `acquireDataSource` and `buildObjectLoader`.

`acquireDataSource(RiskAnalyzer)` returns an instance of the `RiskAnalyzerDataSource` required to run your process. The `RiskAnalyzer` parameter contains context information that the `RunMode` can use to instantiate the data source object.

`buildObjectLoader(RiskAnalyzer)` returns an instance of the `AbstractRiskAnalyzerLoader` required to run your process. The `RiskAnalyzer` parameter contains context information that the RunMode can use to instantiate the object loader.

When implementing `RunMode`, it is critical that your object loader and data source are compatible, meaning that the data source you return produces the specific type of loadable object that your object loader expects.

The method `chooseStartDateRange(VCryptDataAccessMgr, RunSession)` method is used to determine the start date range for your OAAM Offline job. All of your implementors of `RunMode` have a default implementation of this method. The default behavior is as follows. If this is the first time the job has run, you return the start date from the run session's session set if any, or an arbitrary date guaranteed to be earlier than the earliest date in your data source if your session set has no begin date. If this is a resumed job, then you determine, in an implementation specific way, which record you need to start from when the job is resumed.

22.5.1 Extending AbstractLoadLoginsRunMode

This is the appropriate choice if you are loading login data, and you need a custom data source. You must implement the `acquireDataSource(RiskAnalyzer)` method, and return a new instance of your custom data source. If you need a custom

implementation of AbstractRiskAnalyzerLoader, you can override buildObjectLoader(RiskAnalyzer) to return it.

AbstractLoadLoginsRunMode implements the logic to determine the login date at which to resume as follows. The superclass method retrieveLowerBoundDateFromQuery calls an abstract method buildQueryToRetrieveLowerBound, which returns a BharosaDBQuery. The implementation of buildQueryToRetrieveLowerBound in this class selects the most recent VCryptTrackerUserNodeLog.createTime.

Depending on your requirements, you might need to override that behavior. You could override buildQueryToRetrieveLowerBound to add additional criteria to the query or replace the entire query. The only requirement is that the query return a single Date type result. You could instead override the retrieveLowerBoundDateFromQuery or chooseStartDateRange methods, to replace or extend the algorithm.

22.5.2 Extending AbstractLoadTransactionsRunMode

This is the appropriate choice if you are loading transactional data, because you will need a custom data source. You must implement the acquireDataSource(RiskAnalyzer) method, and return a new instance of your custom data source. If you need a custom implementation of AbstractRiskAnalyzerLoader, you can override buildObjectLoader(RiskAnalyzer) to return it.

AbstractLoadTransactionsRunMode implements the logic to determine the login date at which to resume as follows. The superclass method retrieveLowerBoundDateFromQuery calls an abstract method buildQueryToRetrieveLowerBound, which returns a BharosaDBQuery. The implementation of buildQueryToRetrieveLowerBound in this class selects the most recent VTransactionLog.createTime.

Depending on your requirements, you might need to override that behavior. You could override buildQueryToRetrieveLowerBound to add additional criteria to the query or replace the entire query. The only requirement is that the query return a single Date type result. You could instead override the retrieveLowerBoundDateFromQuery or chooseStartDateRange methods, to replace or extend the algorithm.

22.5.3 Extending PlaybackRunMode

This is the appropriate choice if you have requirements that make it necessary to replace the default playback data source or processing behavior. There are no abstract methods to be implemented, but you can override superclass methods to fulfill your requirements.

If you need a custom data source, you can override acquireDataSource(RiskAnalyzer) to return it. If you need a custom implementation of AbstractRiskAnalyzerLoader, you can override buildObjectLoader(RiskAnalyzer) to return it.

PlaybackRunMode implements the logic to determine the login date at which to resume as follows. The chooseStartDateRange method picks the most recent date out of the following choices, the session set's start date if not null, the run session's last processed date if not null, and arbitrary date guaranteed to be earlier than the earliest date in your data source. The third option will only be chosen if the first two are null.

Part VII

Troubleshooting

Part VII contains the following chapter:

- Chapter 23, "FAQ/Troubleshooting"

23

FAQ/Troubleshooting

This chapter provides troubleshooting tips and answers to frequently asked questions. It contains the following sections:

23.1 Techniques for Solving Complex Problems

This section describe a process to enable you to more easily solve a complex problem. It contains the following topics:

23.1.1 Simple Techniques

You can work your way through some simple troubleshooting techniques to try to solve a problem.

Steps	Description
Experience	You have seen this problem before or it is simply something you know the answer to.
Post to the Forum	This is not the first step. Only valid once basics have been applied and a second opinion is needed. Appropriate during rigorous analysis, but not before.

Steps	Description
Intuitive leap (or guess)	The problem just inspires a guess at a cause. You have a feel for the problem or rather its cause. This can be very effective and result in a quick resolution, but without proper confirmation, it often leads to the symptom being fixed and not the real cause being resolved.
Review basic diagnostics	Check the logs for errors and the flow. Check flow (HTTP headers, network packet trace, SQL trace, strace). Run through and document the flow. Cross check with configuration details to ensure flow is expected.
Read the error message	Reading the error and the flow information will give a big clue. Taken together with some knowledge of the way the component works, this can give a lot of insight. Always check knowledge (Oracle and search engine) for matches. Perform any diagnostics needed to establish if the error is key. With multiple errors, look to see which is likely the cause and which are just consequences.
Compare	Compare the logs and flows with a working system. Perform a test case. If it happens only at a certain site, then compare the differences.
Divide	Break the problem down

23.1.2 Divide and Conquer

Steps to reduce the problem to a manageable issue are listed in this section.

Process	Description
Simplify the problem	Make a problem as simple as possible.
Remove components that are not needed	Most problems involve complex components and connections between them. Most involve third party components. So where ever possible, eliminate third party components first and then as many components and custom components as possible (for example, command line not application, SQLPLUS is not an application.)?
Reduce complexity	Test to see if a simpler version of the problem exists with the same symptoms. (for example, remove components of a complex Select, or a search filter, check if a single request or few requests will suffice)?.
Like fixing an underground pipe with a leak	Imagine a complex configuration as being a underground hose pipe with a leak. You know something is wrong, there is a leak someplace, but not where it is.
List the components	Draw a box for each components and a line where it is connected to the next. Note the protocols used to join them.
Check both ends	What goes in should come out the same. If you see data in and out results in a problem then it is one of the ends that is wrong. If the flow is not as expected the problem is in between.
Lazy Y	Test points in the configuration to find where the deviation occurs. Once established (beyond doubt) that a piece of the configuration behaves as expected it can be ignored.
Repeat	Repeat this loop to close in on the problem
Help	When 3rd party components are involved in the issue, get help from the others and work on the issue together.

23.1.3 Rigorous Analysis

All or part of the process should be applied if:

- a problem is complex
- a problem is highly escalated
- a problem was not solved with the first attempts

- a problem is getting out of control
- a problem has potential for getting out of control

23.1.4 Process Flow of Analysis

The process flow of analysis is presented below:

1. State the problem.
2. Specify the problem.

   Develop possible causes from:

   a. Knowledge and experience

   b. Distinctions and changes
3. Test possible causes against the specification.
4. Determine most probable cause.
5. Verify the solution.

23.1.4.1 State the Problem

Stating the problem is the most important step to solving the issue.

Step	Description
Ensure a clear and concise problem statement	Stating the problem is the most important step. It is the most commonly ignored or at least the problem statement is assumed. It is pointless trying to solve a problem until the problem statement is stated. Otherwise what are you actually trying to fix? If you do not know what it is you are fixing how can you fix it?
Consider if the problem stated can be explained	If so, then it is not the problem statement --If the problem statement can be explained then back up and try and get a more correct problem statement. This is a case to start communicating if you are helping someone solve his problem. Either ask some direct questions to narrow down the issue or just pick up the telephone and talk to the person to clarify the real issue. If there are lots of issues then start noting them down as separate issues.
Do not settle for a vague statement	Vague problem statements, like "bad performance", "something crashes" are of no use and commonly are the cause for issues to be long running and out of control.
Never combine problems in a single statement	Ensure there is only one problem being dealt with. Do not accept combined problems. The combined problem is either multiple distinct problems or some of the problems are actually symptoms.

23.1.4.2 Specify the Problem

Describe problems in detail and ask focused questions to gather pertinent information.

Step	Description
Specify the problem	These are symptoms of the problem.
Start by asking questions	Ask questions such as What, Where, When, and to what Extent?
What?	**What** tends to be the obvious question and is mostly a list of facts and symptoms; what deviated from the expectation?
Where?	**Where** may or may not be relevant, but is worth asking as it is often significant and often overlooked.

Step	Description
When	**When** is very important as time lines helps identify patterns and establish what change triggered the problem.
Extent	**Extent** or **how many** is particularly useful in establishing probable causes. If it is all the systems for example then check if it affects all systems or try a testcase. How often is also important. Once a week is quite different from many times every second and tells us much about the type of issue to look for.
List the symptoms and facts	List the symptoms and facts and how they are significant
What changed?	Something changed that is certain unless the problem has always been there. This is a special case.
Assumptions	Verify the data provided and check for conflicts and contradictions. Always check for any assumptions. Be careful to identify any information that is not verified and thus is only assumed. In fact this is particularly a mistake made by analysts that have more technical experience. Though also occurs a lot when inexperienced analysts are given details from people they perceive as having more knowledge. However trivial an assumption seems, always look for proof and confirmation.

23.1.4.3 What It Never Worked

If the component did not work before, performing these steps:

Considerations	Description
Consider behavior and expectation if performance issue	For cases when the issue is about something that never worked correctly the first issue is to establish what correct behavior really is and if it is reasonable? This also allows us to set proper expectations from the outset. This is especially true for performance issues.
Confirm that there is no misunderstanding	Establish that the requirement is reasonable.
Do not compare Apples with Oranges	Agree on a specific goal. Focus on that issue only.
Consider all components involved	Consider all components involved: ■ Not just the software ■ Hardware is fast enough?
Consider if the solutions is just to change perception	What can you see that causes you to think there's a problem? ■ Human factors ■ Perception

23.1.4.4 IS and IS NOT but COULD BE

Consider what the problem is, what it isn't, and what it could be.

Step	Description
IS and IS NOT but COULD BE	For every fact or symptom ask this question: IS and IS NOT but COULD BE

Step	Description
Provide comparison	A test case often is the key to establishing something to compare the problem with. If it reproduces the issue then it does not help the problem analysis as such, but it is extremely useful when passing the problem to the next team to work on the fix. It also allows quicker testing of potential fixes and solutions (workarounds), not to mention you would be gaining experience.
If there is no comparison, create a test case	If it does not reproduce then it provides something to compare the problem system with and perhaps even a possible work around.

23.1.4.5 Develop Possible Causes

Problem solving involves developing possible causes.

Development	Description
Knowledge and experience	You can use your knowledge and experience to recognize possible causes ■ Seen before ■ Seen it in the documentation ■ Support note or through search engine
Distinctions and changes	You can make a list of distinctions and changes to narrow down causes: ■ Only at this site or on one platform ■ Just after upgrade ■ When load increased ■ Only on Thursdays
Examine each of the symptoms and comparisons	Consider each of the facts and ensure that they are relevant and that they are not conflicting

23.1.4.6 Test Each Candidate Cause Against the Specification

Test each candidate cause against the specification:

- Each possible cause must fit all the items in the specification
- If you end up with no causes then go back and refine the process
- Causes must explain both the IS and the IS not but COULD be
- Determine the most probable cause
- Do not discount any causes that fit

23.1.4.7 Confirm the Cause

Confirm the cause so that you can devise an action plan.

You can:

- Devise ways to test the possible causes
- Observe
- Test assumptions

- Experiment
- Test solution and monitor

The main point here is to devise action plans to prove or disprove the theories. It is important to communicate the reason for each action plan. Especially when asking for a negative test, i.e. a test that is to prove something is not true. People might assume all action plans are attempts to solve the problem and resist any thing they think is not directed in the direction.

23.1.4.8 Failures

When one solution fails, just start back at the beginning and apply the approach once again, updated with the new results. Really complex problems will often take several iterations.

The process is not infallible.

Main causes of failure are:

- Poor or incorrect problem statement
- Inaccurate or vague information
- Missing the key distinctions in IS vs. IS NOT
- Allowing assumptions to distort judgment
- Not involving a broader set of skills

23.2 Troubleshooting Tools

This section contains information about tools and processes you can use to investigate and troubleshoot issues with your system.

Table 23–1 lists the general and OAAM-specific tools you can use for troubleshooting problems.

Table 23–1 Troubleshooting Tools

Category	Description
General Tools	■ Middleware Enterprise Manager ■ Database Enterprise Manager ■ Monitor Data in DMS ■ Audit Data ■ Ping/Network Check Tools
OAAM Specific Tools	■ Dashboard ■ Monitor Data ■ Log files

Table 23–2 provides items to check for when troubleshooting the system.

Table 23–2 Troubleshooting Tips

Tips	Reason
Check the operating system	Some issues may be platform specific. For example, Java keystores created on non-IBM platforms will not work on IBM platforms
Check WebLogic Server version	Make sure OAAM is installed on a WebLogic server certified for 11g
Check the JDK (Sun or JRockit)	Make sure the JDK is certified for the Identity Management 11g Suite
Change logging configuration through Enterprise Manager	Make sure the log level is changed appropriately before tracing and debugging
Search for log messages through Enterprise Manager	Log messages record information you deem useful or important to know about how a script executes.
Use the Execution Context ID to search for log messages	The ECID is a unique identifier that can be used to correlate individual events as being part of the same request execution flow.
Use the WebLogic Console to monitor database connection pool	Check the health of the connection pool through the WebLogic Console.

Table 23–3 summarizes problems and the checks you can perform to troubleshoot and solve the problem.

Table 23–3 Problems and Tips

Problem	Checks You Can Perform
Common Troubleshooting Use Cases	■ Most of the operations are slow ■ Server is throwing out of memory exceptions ■ Server is throwing encryption related exceptions ■ Connection pool related errors occur when starting the server ■ Errors while starting managed servers after upgrade from 11.1.1.4 to 11.1.1.5 ■ OAAM CLI script issues ■ SOAP call issues ■ Native integration issues
Most of the Operations are Slow	■ Check performance of OAAM policies – Use the dashboard to see the performance of the rules – Tune rules or their parameterd if necessary ■ Check the database using Enterprise Manager and see if there are any queries that are slow. Follow Enterprise Manager recommendation to add suggested indexes ■ Check if the application server CPU is high Take a thread dump if possible ■ Check the connectivity and network speed between application server and database ■ Use the IP of the database machine in data source settings
Server is Throwing Out of Memory Exceptions	■ Check the configuration of the OAAM WebLogic Domain ■ See if all the OAAM web applications are deployed on the same managed servers ■ Increase the heap size of the managed server
Connection Pool Errors	■ Make sure the database listener is running ■ Use IP address rather than name in JDBC URL ■ Make sure the database service name is correct ■ Make sure the connection pool is not too "large" Check if there are too many managed servers accessing the same database
Errors While Starting the Managed Server After Upgrade	■ Make sure encryption keys are properly copied ■ Make sure all manual steps are followed that are in the upgrade documentation ■ Check the WebLogic Console and make sure all web applications are targeted properly to their managed servers

Table 23–3 (Cont.) Problems and Tips

Problem	Checks You Can Perform
OAAM CLI Script Issues	
	■ Make sure the JAVA_HOME environment variable is set to the JDK certified for the Identity Management Suite for 11g ■ Make sure CLI related properties are set in the `oaam_cli.properties` file.
SOAP Call Issues	
	■ Known issues exist with time-outs in SOAPGenericImpl ■ OWSM is enabled by default, so you need to set OWSM policy before using SOAP ■ Make sure the SOAP server URL including the port number is valid
Native Integration Issues	
	■ Make sure the appropriate version of the OAAM Extensions Shared Library is used (the WAR should use the war version and EAR should use the ear version ■ Make sure the OAAM data source is created and the JNDI name is correct (it should match the JNDI name of the OAAM Server) ■ Make sure the native application is using the same keys that are used by the OAAM Admin and OAAM server ■ Issues with the encryption keys – Make sure all the managed servers are on the same WebLogic domain or copy the keys across the domains – If using non-11g servers, use the Java keystores ■ Shared library usage by many applications on the same server Currently the OAAM Extensions Shared Library cannot be used by more than one application on the same managed server

23.3 OAAM UIO Proxy

UIO ISA Proxy

To troubleshoot the OAAM UIO 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 the 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 the OAAM Server rule you must ensure that:
  - 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 UIO Proxy, enable tracing to a file and set the trace level to 0x8008f. Doing so wil print detailed interceptor evaluation and execution information to the log file.

UIO Apache Proxy

Tips to troubleshoot problems with the UIO Apache Proxy 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 UIO Apache Proxy log.
- When migrating ISA configuration XML to be used with the UIO Apache 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/libxml.en.html

     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 UIO Apache Proxy distribution and run following command:

     ```
     xmllint --noout --relaxng UIO_Config.rng <your config xml file>
     ```

     And fix any errors that are reported.

- The UIO Apache Proxy 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--user name, 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 UIO 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 UIO 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 10*g* or Oracle Adaptive Access Manager with a listener.

No Changes in Proxy in 11*g*

Question/Problem: Are there changes between 10*g* and 11*g* for the UIO Proxy?

Answer/Solution: There has been no changes in the proxy between 10g and 11*g*. There is no dependency on OHS etc. The user has to use Apache 2.2.8 only.

Adding appid to HTTP Headers

Question/Problem: In `TestConfig.xml`, should we be adding `appid` to HTTP headers for both the PSFT URLs and the `/asa/` URLS?

Answer/Solution: No, just to the `/asa/` URLs. It should be adding the `app-id` to only the `/asa/` URLs, not needed for `PSFT` urls.

Contains Match

Question/Problem: Should a condition with "contains" match if there is an exact match?

Answer/Solution: Yes.

Request URL

Question/Problem: Can request URL be a partial URL? (Such as just first part of URL?)

Answer/Solution: No, URL must be an exact match and query parameters, such as anything after a "?" are not considered part of the URL, so they would have to be trapped with a condition, and not included as part of the URL.

23.4 Knowledge-Based Authentication

Prompt a User with Two Challenge Questions

Question/Problem: I would like to prompt a user with two challenge questions when they attempt to logon from a new device. How can this be achieved given that the questions are randomly picked, raising the possibility that the same question may be displayed twice?

Answer/Solution: The OAAM "one question at a time" flow is by design. It is better security practice to present one question and only show the next question once the user has successfully answered the challenge. This protects the questions from being harvested for use in a phishing exercise. As well, OAAM allows users to have multiple attempts at a question which entails keeping track of how many wrong answers they have entered. If there were more than one question displayed at a time it would be difficult to maintain and possibly confusing to end users. If you want to challenge a user with more than one question you should do so by presenting them in separate sequential screens. OAAM does not support authentication of more than one question at a time.

23.5 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.

You will need to modify `bharosa_server.properties`.

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 `bharosa_server.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.

Can OAAM Restrict the Number of Devices used by a User

Question/Problem: Is there any way to configure the limit for a user to use fewer number of devices, such as 5 or 6 and block any access from the devices which are not in the configured list for specifc user ?

Answer/Solution: For usability and security reasons OAAM does not support limiting a user to a set number of devices. As well, this behavior is not required for proper security coverage since OAAM profiles the behavior of users including the devices they use. The total number of devices is not a good measure of risk as some end users may utilize many devices as part of their normal behavior. Instead OAAM keeps track of how often a user utilizes a specific device, who else has used that same device in the past and with what frequency. These evaluations can better assess the level of risk associated with an access request.

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 or another form.) 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 in to `bharosa_server.properties`:

  ```
  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.

23.6 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 7, "Customizing Oracle Adaptive Access Manager."

23.7 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.

23.8 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

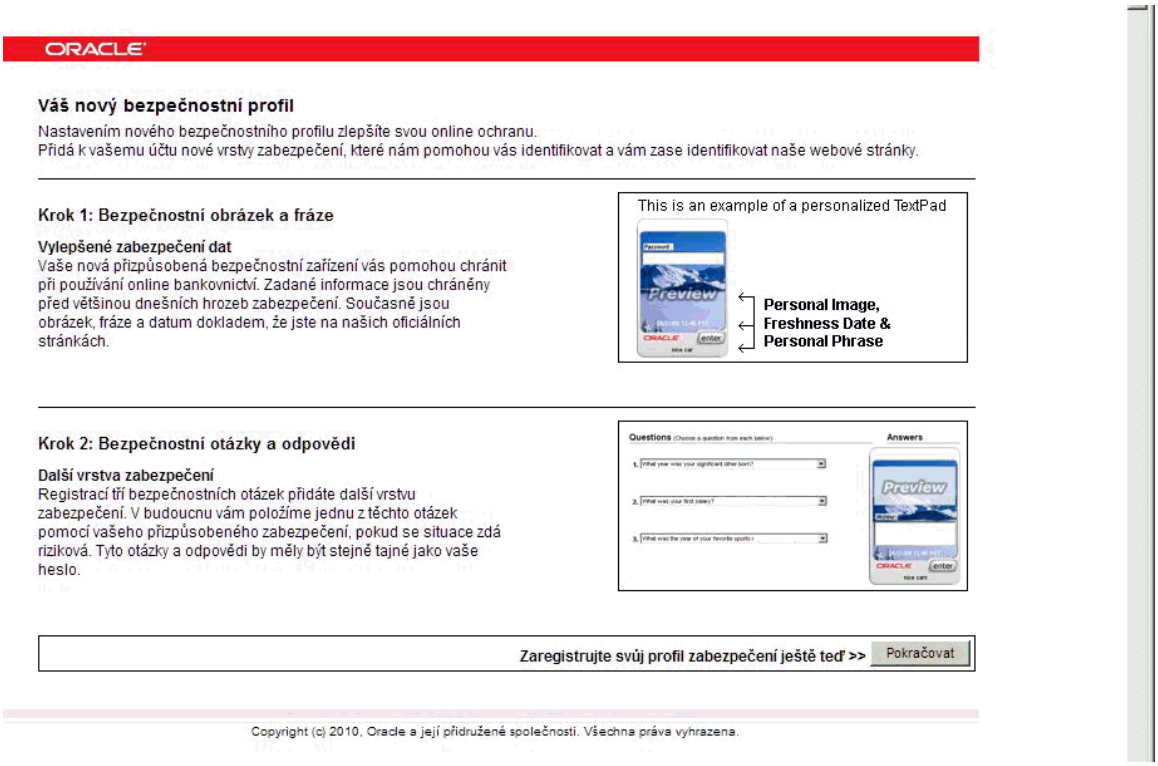
The frame and key samples are provided in English only. Master files for the virtual authentication device frames and keys along with descriptions of the parts are provided on request. You may create your own custom frame and key images and deploy them using product documentation. Any and all alterations 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.



Answer/Solution: Globalized virtual authentication device image files including the authentication registration flows are not provided. The deployment team develop these.

23.9 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.

23.10 Failure Counter

For the auto failure counter increment to work, Client Type for updateAuthStatus must be set to 9 (Question/Answer).

Part VIII

Glossary

This part contains the glossary.

Glossary

Access Authentication

In the context of an HTTP transaction, the basic access authentication is a method designed to allow a web browser, or other client program, to provide credentials – in the form of a user name and password – when making a request.

Action

Rule result which can impact users such forcing them to register a security profile, KBA-challenging them, blocking access, asking them for PIN or password, and so on.

Adaptive Risk Manager

A category of Oracle Adaptive Access Manager features. Business and risk analytics, fraud investigation and customer service tools fall under the Adaptive Risk Manager category.

Adaptive Strong Authenticator

A category of Oracle Adaptive Access Manager features. All the end-user facing interfaces, flows, and authentication methods fall under the Adaptive Strong Authenticator category.

Alert

Rule results containing messages targeted to specific types of Oracle Adaptive Access Manager users.

API

An Application Programming Interface defines how to access a software-based service. 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.

Attribute

Attributes are the particular pieces of information associated with the activity being tracked. An example is the time of day for a login. Patterns collect data about members. If the member type is **User**, the pattern will collect data about users.

Authentication

The process of verifying a person's, device's, application's identity. Authentication deals with the question "Who is trying to access my services?"

Authentication Status

Authentication Status is the status of the session (each login/transaction attempt creates a new session).

Examples are listed below:

- If a user logs in for the first time and he goes through the registration process, but decides not to complete the registration process and logs out, the authentication status for this user session is set as "Pending Activation."
- If a user logs in from a different device/location, he is challenged. He answers the challenge questions incorrectly in all the three attempts, the authentication status for this session is set as "Wrong Password."
- If a user logs in and is taken to the final transaction page or success page, the authentication status for the particular session is set as "Success."
- If the user is a fraud and is blocked, the status for the session is set as "Block."

Authorization

Authorization regards the question "Who can access what resources offered by which components?"

Autolearning

Autolearning is a set of features in Oracle Adaptive Access Manager that dynamically profile behavior in real-time. The behavior of users, devices and locations are recorded and used to evaluate the risk of current behavior.

Black List

A given list of users, devices, IP addresses, networks, countries, and so on that are blocked. An attack from a given member can show up on a report and be manually added to a blacklist at the administrator's discretion.

Blocked

If a user is "Blocked," it is because a policy has found certain conditions to be "true" and is set up to respond to these conditions with a "Block Action." If those conditions change, the user may no longer be "Blocked." The "Blocked" status is not necessarily permanent and therefore may or may not require an administrator action to resolve. For example, if the user was blocked because he was logging in from a blocked country, but he is no longer in that country, he may no longer be "Blocked."

Bots

Software applications that run automated or orchestrated tasks on compromised PCs over the internet. An organization of bots is known as a bot net or zombie network.

Browser Fingerprinting

When the user accesses the system, OAAM collects information about the computer. By combining all that data, the site creates a fingerprint of the user's browser. This fingerprint could potentially uniquely identify the user. Information gathered that makes up the browser fingerprint include the browser type used, plug-ins installed, system fonts, and the configuration and version information from the operating system, and whether or not the computer accepts cookies.

The browser and flash fingerprints are tracked separately. The fingerprints are available in the session listing and details pages and you can get further details about the fingerprint by opening the respective details pages. Hence, you can have both

fingerprints available, but if the user has not installed flash then the digital fingerprint (flash) is set to null.

Cache Data

Information about historical data during a specified time frame

Case

Cases provide tools to track and solve customer service issues.

A **case** is a record of all the actions performed by the CSR to assist the customer as well as various account activities of the customer. Each case is allocated a **case number**, a unique case identification number.

Challenge Questions

Challenge Questions are a finite list of questions used for secondary authentication.

During registration, users are presented with several question menus. For example, he may be presented with three question menus. A user must select one question from each menu and enter answers for them during registration. Only one question from each question menu can be registered. These questions become the user's "registered questions."

When rules in OAAM Admin trigger challenge questions, OAAM Server displays the challenge questions and accepts the answers in a secure way for users. The questions can be presented in the QuestionPad, TextPad, and other pads, where the challenge question is embedded into the image of the authenticator, or simple HTML.

Challenge Type

Configuration of a type of challenge (ChallengeEmail, ChallengeSMS, ChallengeQuestion)

Checkpoint

A checkpoint is a specified point in a session when Oracle Adaptive Access Manager collects and evaluates security data using the rules engine.

Examples of checkpoints are:

- Pre-authentication - Rules are run before a user completes the authentication process.
- Post-authentication - Rules are run after a user is successfully authenticated.

Configurable Actions

Configurable Actions allow a user to create new supplementary actions that occur after the running of rules.

Completed Registration

Status of the user that has completed registration. To be registered a user may need to complete all of the following tasks: Personalization (image and phrase), registering challenge questions/answers and email/cell phone.

Condition

Conditions are configurable evaluation statements used in the evaluation of historical and runtime data.

Cookie

A cookie is a small string of text or data stored on a user's computer. Oracle Adaptive Access Manager uses two types of cookies to perform device identification. One is the browser cookie (also known as secure cookie) and the other is the flash cookie (also known as digital cookie). The browser cookie value is constructed using the browser user agent string. The flash cookie value is constructed using data from the OAAM flash movie.

CSR

Customer service representatives resolve low risk customer issues originating from customer calls. CSRs has limited access to OAAM Admin

- View the reason why a login or transaction was blocked
- View a severity flag with alert status to assist in escalation
- Complete actions such as issuing temporary allow for a customer

CSR Manager

A CSR Manager is in charge of overall management of CSR type cases. CSR Managers have all the access and responsibilities of a CSR plus access to more sensitive operations.

Dashboard

Provides a real-time view of activity via aggregates and trending.

Data Mining

Data mining is the practice of automatically searching large stores of data to discover patterns and trends that go beyond simple analysis. Data mining uses sophisticated mathematical algorithms to segment the data and evaluate the probability of future events. Data mining is also known as Knowledge Discovery in Data (KDD). Data mining can answer questions that cannot be addressed through simple query and reporting techniques.

Data Type

An attribute of data that represents the kind and structure of the data. For example, String.

Delivery Channel

Delivery mechanism used to send the OTP to the user. Email, SMS, IM, and so on are delivery channels.

Device

A computer, PDA, cell phone, kiosk, etc used by a user

Device Fingerprinting

Device fingerprinting collects information about the device such as browser type, browser headers, operating system type, locale, and so on. Fingerprint data represents the data collected for a device during the login process that is required to identify the device whenever it is used to log in. The fingerprinting process produces a fingerprint that is unique to the user and designed to protect against the "replay attacks" and the "cookie based registration bypass" process. The fingerprint details help in identifying a device, check whether it is secure, and determine the risk level for the authentication or transaction.

A customer typically uses these devices to log in: desktop computer, laptop computer, PDA, cell phone, kiosk, or other web enabled device.

Device Identification

During the registration process, the user is given an option to register his device to the system. If a user tries to login from a registered device, the application knows that it is a safe and secure device and allows the user to proceed with his transactions. This process is also called device identification.

Device Registration

Device registration is a feature that allows a user to flag the device (computer, mobile, PDA, and others) being used as a safe device. The customer can then configure the rules to challenge a user that is not coming from one of the registered devices.

Once the feature is enabled, information about the device is collected for that user. To make use of the information being collected, policies must be created and configured. For example, a policy could be created with rules to challenge a user who is not logging in from one of the registered devices.

encrypted

Information that is made unreadable to anyone except those possessing special knowledge

Entities Editor

A tool to edit entities, a user-defined structure that can be reused across different transactions. Only appropriate and related fields should be grouped into an Entity.

Entity

An entity is a user-defined data structure that can be re-used across different transactions.

Environment

Tools for the configuration system properties and snapshots

Expiration Date

Date when CSR case expires. By default, the length of time before a case expires is 24 hours. After 24 hours, the status changes from the current status to Expired. The case could be in pending, escalated statuses when it expires. After the case expires, the user will not be able to open the case anymore, but the CSR Manager can. The length of time before a case expires is configurable.

Execution Types

Two execution types for configurable actions are listed:

- Synchronous - Synchronous actions are executed in the order of their priority in ascending order. For example, if the user wants to create a case and then send an email with the Case ID, the user would choose synchronous actions. Synchronous actions will trigger/execute immediately.

  If the actions are executing in sequential order and one of the actions in the sequence does not trigger, the other actions will still trigger.

- Asynchronous actions are queued for execution but not in any particular sequence. For example, if you want to send an email or perform some action and do not care about executing it immediately and are not interested in any order of execution, you would choose asynchronous actions.

Enumerations

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
```

Fat Fingering

This algorithm handles Answers with typos due to the proximity of keys on a standard keyboard.

Flash Fingerprinting

Flash fingerprinting is similar to browser fingerprinting but a flash movie is used by the server to set or retrieve a cookie from the user's machine so a specific set of information is collected from the browser and from flash. The flash fingerprint is only information if flash is installed on the client machine.

The fingerprints are tracked separately. The fingerprints are available in the session listing and details pages and you can get further details about the fingerprint by opening the respective details pages. Hence, you can have both fingerprints available, but if the user has not installed flash then the digital fingerprint (flash) is set to null.

Fraud Investigator

A Fraud Investigator primarily looks into suspicious situations either escalated from customer service or directly from Oracle Adaptive Access Manager alerts. Agents have access to all of the customer care functionality as well as read only rights to security administration and BI Publisher reporting.

Fraud Investigation Manager

A Fraud Investigation Manager has all of the access and duties of an investigator plus the responsibility to manage all cases. An Investigation Manager must routinely search for expired cases to make sure none are pending.

Fraud Scenario

A fraud scenario is a potential or actual deceptive situation involving malicious activity directed at a company's online application.

For example, you have just arrived at the office on Monday and logged into OAAM Admin. You notice that there are a high number of logins with the status "Wrong Password" and "Invalid User" coming in from a few users. Some appear to be coming in from different countries, and some appear to be local. You receive a call from the fraud team notifying you that some accounts have been compromised. You must come up with a set of rules that can identify and block these transactions.

Groups

Collection of like items. Groups are found in the following situations

- Groups are used in rule conditions
- Groups that link policy to user groups
- Action and alert groups

HTTP

Hypertext Transfer Protocol

IP address

Internet Protocol (IP) address

Job

A job is a collection of tasks that can be run by OAAM. You can perform a variety of jobs such as load data, run risk evaluation, roll up monitor data, and other jobs.

KBA Phone Challenge

Users can be authenticated over the phone using their registered challenge questions. This option is not available for unregistered users or in deployments not using KBA.

KeyPad

Virtual keyboard for entry of passwords, credit card number, and on. The KeyPad protects against Trojan or keylogging.

Keystroke Loggers

Software that captures a user's keystrokes. Keylogging software can be used to gather sensitive data entered on a user's computer.

Knowledge Based Authentication (KBA)

OAAM knowledge based authentication (KBA) is a user challenge infrastructure based on registered challenge questions. It handles Registration Logic, challenge logic, and Answer Logic.

Location

A city, state, country, IP, Network ID, etc from which transaction requests originate.

Locked

"Locked" is the status that Oracle Adaptive Access Manager sets if the user fails a KBA or OTP challenge. The "Locked" status is only used if the KBA or One Time-Password (OTP) facility is in use.

- OTP: OTP sends a one-time PIN or password to the user through a configured delivery method, and if the user exceeds the number of retries when attempting to provide the OTP code, the account becomes "Locked."

- KBA: For online challenges, a customer is locked out of the session when the Online Counter reaches the maximum number of failures. For phone challenges, a customer is locked out when the maximum number of failures is reached and no challenge questions are left.

After the lock out, a Customer Service Representative must reset the status to "Unlocked" before the account can be used to enter the system.

Malware

Malware is software designed to infiltrate or damage a computer system without the owner's informed consent. Malware may contain key loggers or other types of malicious code.

Man-In-The-Middle-Attack (Proxy Attacks)

An attack in which a fraudster is able to read, insert and modify at will, messages between two parties without either party knowing that the link between them has been compromised

Multifactor Authentication

Multifactor authentication (MFA) is a security system in which more than one form of authentication is implemented to verify the legitimacy of a transaction. In contrast, single factor authentication (SFA) involves only a User ID and password.

Multiprocessing Modules (MPMs)

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.

Mutual Authentication

Mutual authentication or two-way authentication (sometimes written as 2WAY authentication) refers to two parties authenticating each other suitably. In technology terms, it refers to a client or user authenticating himself to a server and that server authenticating itself to the user in such a way that both parties are assured of the others' identity.

Native Integration

Native integration involves customizing the application to include OAAM API calls at various stages of the login process. The application invokes Oracle Adaptive Access Manager directly and the application itself manages the authentication and challenge flows.

- SOAP service wrapper API: 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 application makes SOAP calls to interact with Oracle Adaptive Access Manager.
- Static linking: The processing engine for Oracle Adaptive Access Manager (OAAM Library) is imbedded with the application. It leverages the underlying database directly for processing.

OAAM Admin

Administration Web application for all environment and Adaptive Risk Manager and Adaptive Strong Authenticator features.

OAAM Server

Adaptive Risk Manager and Adaptive Strong Authenticator features, Web services, LDAP integration and user Web application used in all deployment types except native integration

One Time Password (OTP)

One Time Password (OTP) is a form of out of band authentication that is used as a secondary credential and generated at pre-configured checkpoints based on the policies configured.

OTP Anywhere

OTP Anywhere is a risk-based challenge solution consisting of a server generated one time password delivered to an end user via a configured out of band channel. Supported OTP delivery channels include short message service (SMS), eMail, instant messaging and voice. OTP Anywhere can be used to compliment KBA challenge or instead of KBA. As well both OTP Anywhere and KBA can be used alongside practically any other authentication type required in a deployment. Oracle Adaptive Access Manager also provides a challenge processor framework. This framework can be used to implement custom risk-based challenge solutions combining third party authentication products or services with OAAM real-time risk evaluations.

Oracle Adaptive Access Manager

A product to protect the enterprise and its customers online.

Oracle Adaptive Access Manager

- provides multifactor authentication security
- evaluates multiple data types to determine risk in real-time
- aids in research and development of fraud policies in offline environment
- integrates with access management applications

Oracle Adaptive Access Manager is composed of two primary components: OAAM Server and OAAM Admin.

Oracle Data Mining (ODM)

Oracle Data Mining is an option to the Oracle Database EE, provides powerful data mining functionality

Organization ID

The unique ID for the organization the user belongs in

Out Of Band Authentication

The use of two separate networks working simultaneously to authenticate a user. For example: email, SMS, phone, and so on.

Pattern

Patterns are configured by an administrator and record the behavior of the users, device and locations accessing the system by creating a digest of the access data. The digest or profile information is then stored in a historical data table. Rules evaluate the patterns to dynamically assess risk levels.

Personalization Active

Status of the user who has an image, a phrase and questions active. Personalization consists of a personal background image and phrase. The timestamp is generated by

the server and embedded in the single-use image to prevent reuse. Each Authenticator interface is a single image served up to the user for a single use.

Pharming

Pharming (pronounced farming) is an attack aiming to redirect a Web site's traffic to another, bogus Web site.

Phishing

A criminal activity utilizing social engineering techniques to trick users into visiting their counterfeit Web application. Phishers attempt to fraudulently acquire sensitive information, such as user names, passwords and credit card details, by masquerading as a trustworthy entity. Often a phishing exercise starts with an email aimed to lure in gullible users.

PinPad

Authentication entry device used to enter a numeric PIN.

Plug-in

A plug-in consists of a computer program that interacts with a host application (a web browser or an email client, for example) to provide a certain, usually very specific, function "on demand".

Policy

Policies contain security rules and configurations used to evaluate the level of risk at each checkpoint.

Policy Set

A policy set is the collection of all the currently configured policies used to evaluate traffic to identify possible risks. The policy set contains the scoring engine and action/score overrides.

Policy Status

Policy has three status which defines the state of the object or its availability for business processes.

- Active
- Disabled
- Deleted

Deleted is not used.

When a policy is deleted, it is permanently deleted from the database.

By Default every new policy created has status as "Active."

Every copied policy has a default status as "Disabled."

Post-Authentication

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

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

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.

Predictive Analysis

Predictive analytics encompasses a variety of techniques from statistics, data mining and game theory that analyze current and historical facts to make predictions about future events.

Questions Active

Status of the user who has completed registration and questions exists by which he can be challenged.

Question Set

The total number of questions a customer can choose from when registering challenge questions.

QuestionPad

Device that presents challenge questions for users to answer before they can perform sensitive tasks. This method of data entry helps to defend against session hijacking.

Registration

An enrollment process wherein the customer registers challenge questions, secret images, text phrases, one-time passwords, and so on for another layer of security in addition to the login process.

Registered Questions

A customer's registered questions are the questions that he selected and answered during registration or reset. Only one question from each question menu can be registered.

Registration Logic

The configuration of logic that governs the KBA registration process.

Risk Score

The numeric risk level associated with a checkpoint.

Rule Conditions

Conditions are the basic building blocks for security policies.

Rules

Rules are a collection of conditions used to evaluate user activity.

Scores

Score refers to the numeric scoring used to evaluate the risk level associated with a specific situation. A policy results in a score.

Scoring Engine

Oracle Adaptive Access Manager uses scoring engines to calculate the risk associated with access requests, events, and transaction.

Scoring engines are used at the policy and policy set levels. The Policy Scoring Engine is used to calculate the score produced by the different rules in a policy. The Policy Set Scoring Engine is used to calculate the final score based on the scores of policies.

Where there are numerous inputs, scoring is a able to summarize all these various points into a score that decisions can be based on.

Security Token

Security tokens (or sometimes a hardware token, hard token, authentication token, USB token, cryptographic token) are used to prove one's identity electronically (as in the case of a customer trying to access their bank account). The token is used in addition to or in place of a password to prove that the customer is who they claim to be. The token acts like an electronic key to access something.

Severity Level

A marker to communicate to case personnel how severe this case is. The severity level is set by whomever creates the case. The available severity levels are High, Medium, and Low. If a customer suspects fraud, then the severity level assigned is "High." For example, if the customer wants a different image, then the severity level assigned is "Low." Severity levels of a case can be escalated or deescalated as necessary.

Session Hijacking

The term Session Hijacking refers to the exploitation of a valid computer session - sometimes also called a session key - to gain unauthorized access to information or services in a computer system

Snapshot

A snapshot is a zip file that contains Oracle Adaptive Access policies, dependent components and configurations for backup, disaster recovery and migration. Snapshots can be saved to the database for fast recovery or to a file for migration between environments and backup. Restoring a snapshot is a process that includes visibility into exactly what the delta is and what actions will be taken to resolve conflicts.

SOAP

SOAP, originally defined as Simple Object Access Protocol, is a protocol specification for exchanging structured information in the implementation of Web Services in computer networks. It relies on Extensible Markup Language (XML) as its message format, and usually relies on other Application Layer protocols (most notably Remote Procedure Call (RPC) and HTTP) for message negotiation and transmission. SOAP can form the foundation layer of a web services protocol stack, providing a basic messaging framework upon which web services can be built.

Social Engineering

Social engineering is a collection of techniques used to manipulate people into performing actions or divulging confidential information to a fraudulent entity.

Spoofing Attack

In the context of network security, a spoofing attack is a situation in which one person or program successfully masquerades as another by falsifying data and thereby gaining an illegitimate advantage.

Spyware

Spyware is computer software that is installed surreptitiously on a personal computer to intercept or take partial control over the user's interaction with the computer, without the user's informed consent.

Strong Authentication

An authentication factor is a piece of information and process used to authenticate or verify the identity of a person or other entity requesting access under security constraints. Two-factor authentication (T-FA) is a system wherein two different factors are used in conjunction to authenticate. Using two factors as opposed to one factor generally delivers a higher level of authentication assurance.

Using more than one factor is sometimes called strong authentication.

Temporary Allow

Temporary account access that is granted to a customer who is being blocked from logging in or performing a transaction.

TextPad

Personalized device for entering a password or PIN using a regular keyboard. This method of data entry helps to defend against phishing. TextPad is often deployed as the default for all users in a large deployment then each user individually can upgrade to another device if they wish. The personal image and phrase a user registers and sees every time they login to the valid site serves as a shared secret between user and server.

Transaction

A transaction defines the data structure and mapping to support application event/transaction analytics.

Transaction Data

Data that is an abstract item or that does not have any attributes by itself, does not fit into any entity, which exists or is unique by itself is defined as transaction data.

Items that cannot fall into an entity are classified as standalone data.

A classic example is amount or code.

Transaction Definition

Application data is mapped using the transaction definition before transaction monitoring and profiling can begin. Each type of transaction Oracle Adaptive Access Manager deals with should have a separate transaction definition.

Transaction Key

This key value is used to map the client/external transaction data to transactions in the Oracle Adaptive Access Manager Server.

Trigger

A rule evaluating to true.

Transaction Type

The Transaction Definitions that have been configured in this specific installation such as authentication, bill pay, wire transfer, and others.

Trojan/Trojan Horse

A program that installs malicious software while under the guise of doing something else.

User

A business, person, credit card, etc that is authorized to conduct transactions.

Validations

Answer validation used in the KBA question registration and challenge process

Virtual Authentication Devices

A personalized device for entering a password or PIN or an authentication credential entry device. The virtual authentication devices harden the process of entering and transmitting authentication credentials and provide end users with verification they are authenticating on the valid application.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W