Oracle® Fusion Middleware

Developer's Guide for Oracle Adaptive Access Manager

11*g* Release 2 (11.1.2.1)

E60534-02

February 2015

Documentation to help integrators and developers integrate OAAM natively with customer applications by using In-Proc or SOAP mode; perform key customizations for Virtual Authentication Devices (VAD), user action and flows, layout, and headers and footers; and perform custom development.

ORACLE

Oracle Fusion Middleware Developer's Guide for Oracle Adaptive Access Manager, 11*g* Release 2 (11.1.2.1)

E60534-02

Primary Author: Priscilla Lee

Contributors: Niranjan Ananthapadmanabha, Mandar Bhatkhande, Sunil Kumar Joshi, Daniel Joyce, Karthik Kandasamy, Wei Jie Lee, Derick Leo, Paresh Raote, Kuldeep Shah, Nandini Subramani, Elangovan Subramanian, Vidhya Subramanian, Dawn Tyler, Sachin Vanungare, and Saphia Yunaeva.

Contents

5 Creating, Updating, and Searching for Entities Using the Entity API

23 Developing a Custom Loader for OAAM Offline

24 Creating OAAM Oracle BI Publisher Reports

29 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 that have purchased support 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 2 (11.1.2) documentation set:

- *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*
- *Oracle Fusion Middleware Java API Reference for Oracle Adaptive Access Manager*

- *Oracle Fusion Middleware Installation Planning Guide for Oracle Identity and Acess Management*
- *Oracle Fusion Middleware Installation Guide for Oracle Identity and Access Management*
- *Oracle Fusion Middleware Integration Guide for Oracle Identity Management Suite*
- *Oracle Fusion Middleware Enterprise Deployment Guide for Oracle Identity Management*
- *Oracle Fusion Middleware Upgrade and Migration Guide for Oracle Identity and Access Management*
- *Oracle Fusion Middleware High Availability Guide*
- *Oracle Fusion Middleware Administrator's Guide*
- *Oracle Fusion Middleware Performance and Tuning Guide*
- *Oracle Fusion Middleware Administrator's Guide for Oracle Access Management*
- *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*
- *Oracle Fusion Middleware Third-Party Application Server Guide for Oracle Identity and Access 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 integration 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 and manage configuration changes in integrated deployments of Oracle Adaptive Access Manager.

Information in this book is grouped into the following main parts:

- Part I - Native integration
- Part II - Universal Installation Option Proxy
- Part III - Customization and extensions
- Part IV - Oracle Adaptive Access Manager Integration
- Part V - Custom development
- Part VI - Lifecycle management
- Part VII - Troubleshooting tips/FAQ
- Part VIII - Glossary

Detailed information about Oracle Adaptive Access Manager integration with Oracle Identity Manager and Oracle Access Management Access Manager is not covered in this guide. For in-depth conceptual and procedural information, see *Oracle Fusion Middleware Integration Guide for Oracle Identity Management Suite*.

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 Oracle Adaptive Access Manager"
- Chapter 3, "Integrating Native .NET Applications"
- Chapter 4, "Natively Integrating OAAM with Java Applications"
- Chapter 5, "Creating, Updating, and Searching for Entities Using the Entity API"

2

Natively Integrating Oracle Adaptive Access Manager

Oracle Adaptive Access Manager can be natively integrated with an application to provide high performance and highly customizable security. A native integration embeds OAAM in-process inside the protected applications. The application invokes the Oracle Adaptive Access Manager APIs directly to access risk and challenge flows.

This chapter contains guidelines to integrate Oracle Adaptive Access Manager into a client application using the APIs the server exposes. In addition to this *Developer's Guide*, API documentation generated by the Javadoc tool is available. This documentation is provided as HTML and can also be downloaded from the Identity Management Documentation library in HTML format as the *Oracle Fusion Middleware Java API Reference for Oracle Adaptive Access Manager*.

This chapter contains the following sections:

2.1 About OAAM Native Integration

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, challenge the user to provide one-time password, and generate virtual authentication devices such as KeyPad, TextPad, or QuestionPad.

2.1.1 What is Native Integration?

OAAM native integration involves customizing the client application to include OAAM API calls at various stages of the login process. In native integration, the client application invokes Oracle Adaptive Access Manager directly and the application itself manages the authentication and challenge flows.

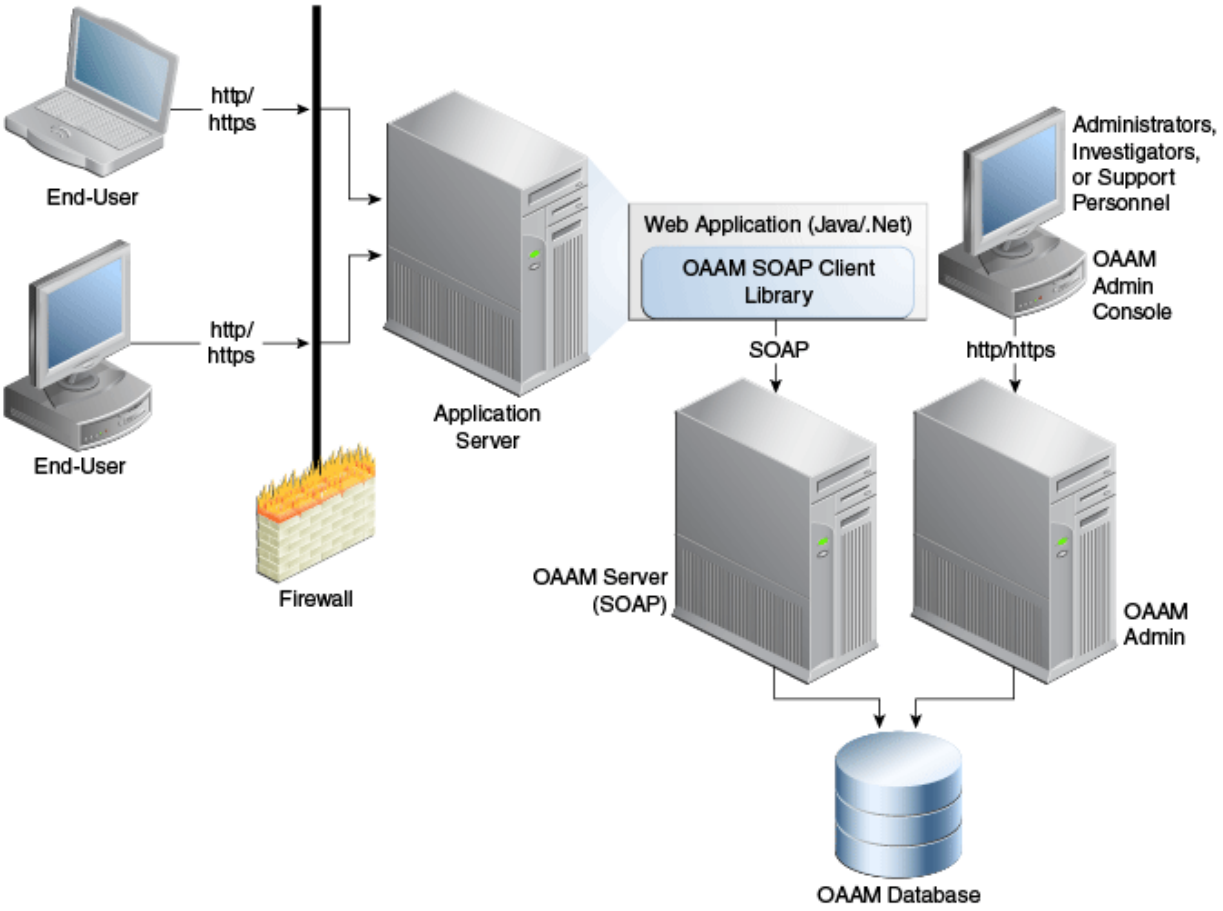
The Native API options are:

- SOAP service wrapper API for Java or .NET applications.
- In-Proc (Link libraries statically) for Java applications only

2.1.2 SOAP Service Wrapper API Integration

Figure 2–1 shows the SOAP service wrapper API integration scenario in which the application communicates with Oracle Adaptive Access Manager using the OAAM Native Client API (SOAP service wrapper API). The application can also communicate with OAAM through Web services.

Figure 2–1 SOAP Service Wrapper API integration Scenario

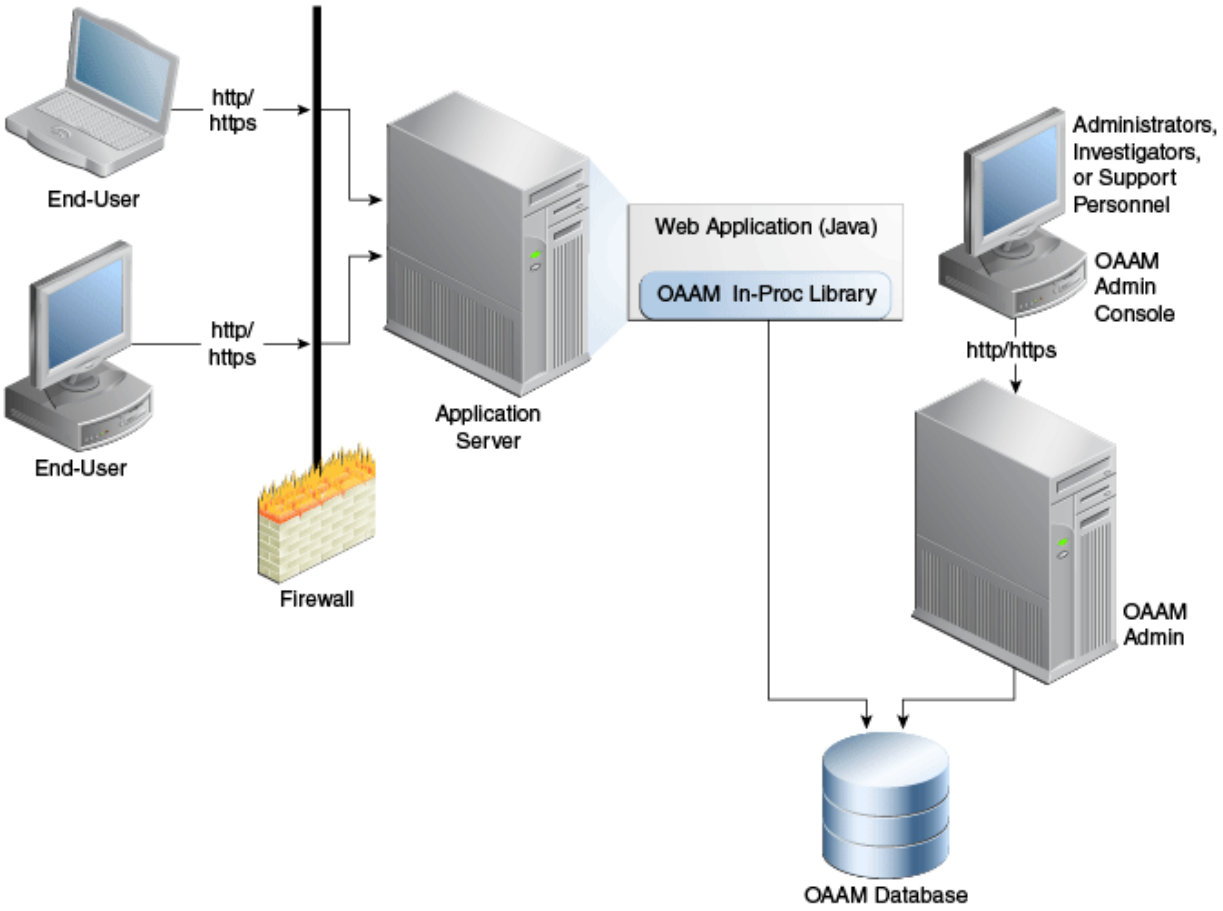


The SOAP service wrapper API enables you to create SOAP objects and invoke SOAP calls and abstracts the SOAP Web Service Definition Language (WSDL) and other Web services details from the application code. Libraries for this API are available for Java, .NET, and C++. This integration requires adding lightweight client libraries (JAR or DLL files) to the client library.

2.1.3 In-Proc Integration

Figure 2–2 shows the In-Proc integration scenario which only involves local API calls and therefore no remote server risk engine calls (SOAP calls).

Figure 2–2 In-Proc integration Scenario Using Local API 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 JAR files and configured properties, as appropriate.

> **Note:** In-Proc integration works only on the WebLogic container where OAAM resides.

2.1.4 SOAP Service Wrapper API vs. In-Proc Method

When communicating with the rules engine, you must decide whether to statically include all the JAR files locally in the application server or to make SOAP calls to a distributed rules engine (typically located on the same host that administers the rules themselves).

Using the SOAP server wrapper 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 are 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.

Even though In-Proc may provide slightly better performance, it is not suitable for all Java clients. In-Proc is recommended for clients developing their own applications with Oracle Adaptive Access Manager built in their J2EE or application.

In-Proc integration 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.

2.1.5 Non-Native Integration - SOAP Services

Using direct SOAP services is preferred if the client does not want to include any of the OAAM client JAR or DLL files within their application. However, to use the adaptive strong authentication functionality, you must use the native Java or .NET integration.

OAAM SOAP services consists of five major modules:

- **VCryptCommon** contains the common APIs.
- **VCryptTracker** contains the APIs for fingerprinting and collecting authentication and transaction logs.
- **VCryptAuth** contains the APIs for accessing the Authenticator and KBA modules.
- **VCryptRulesEngine** contains the APIs for running the rules.
- **VCryptCC** contains the APIs for invoking customer-care-related requests.

2.2 Getting Started

Instructions for setting up the OAAM Sample application are provided in this section. The OAAM Sample application is for demonstration purposes to familiarize you with OAAM APIs. It is not intended to be used as production code since it only provides basic elements of API usage. If you are implementing a native integration, you can develop your application using the OAAM Sample application as a reference. Custom applications developed for these deployments are not supported directly; however, Oracle Support Services can assist you with product issues, such as if you were to encounter problems when using the provided APIs.

2.2.1 Downloading the OAAM Sample Application

The most recent OAAM Sample Application that illustrates Java API integration can be downloaded from My Oracle Support document ID 1351899.1.

Note: This OAAM sample application is different from the one for the task processor framework. Use either OAAM sample applications, but the two may not be deployed together.

If you are interested in task processor integration, see Chapter 22, "Integrating Task Processors".

2.2.2 Setting Up the Native SOAP-based OAAM Sample Application

This section contains instructions to set up the Native SOAP-based OAAM sample application.

2.2.2.1 Pre-requisites

Before you set up the SOAP-based OAAM sample application you need:

- Oracle Adaptive Access Manager to be installed, configured, and running
- Oracle Adaptive Access Manager SOAP service to be enabled and accessible from the host where the OAAM sample application is being deployed
- Details about the database host, username, and password used by Oracle Adaptive Access Manager
- The most recent OAAM Sample Application and related files that illustrate Java API integration can be downloaded from My Oracle Support document ID 1351899.1.
- The latest `webserviceclient+ssl.jar` file.

  A runtime JAR file, called `webserviceclient+ssl.jar` contains the runtime implementation of SSL. If you are not deploying the sample application in a WebLogic container, obtain the file from your Oracle WebLogic Server installation and then, copy it to your client application development computer.

2.2.2.2 Installing and Configuring the OAAM Sample Application

To set up the SOAP-based OAAM sample application, proceed as follows:

1. Create an `oaam_sample` directory.
2. Extract the `oaam_sample_soap.zip` file into the `oaam_sample` directory.
3. Edit the `oaam_custom.properties` file under the *`customer application deployment`*`/WEB-INF/classes` directory:
   - Set the `vcrypt.tracker.soap.url` property.

     `vcrypt.tracker.soap.url=http://`*`host-name`*`:`*`port`*`/oaam_server/services`

     This setting specifies the location of the web services with which the application will communicate. Access to this URL is allowed to the users in the OAAM Web Services Group group.
   - Set the soap class `vcrypt.common.util.vcryptsoap.impl.classname` property.

     The available option is:

     `com.bharosa.vcrypt.common.impl.VCryptSOAPGenericImpl`

     This setting specifies for the application which libraries to use when creating SOAP messages to exchange with the OAAM services.
   - Set `bharosa.image.dirlist` to the absolute directory path where OAAM images are available.
4. Set the following properties using the Properties editor:

```
vcrypt.tracker.impl.classname=
  com.bharosa.vcrypt.tracker.impl.VCryptTrackerSOAPImpl
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.BharosaConfigLoadPropsImpl
vcrypt.tracker.soap.useSOAPServer=true
vcrypt.soap.disable=false
vcrypt.soap.auth.keystoreFile=system_soap.keystore
```

If SOAP Authentication is not enabled, set the following property:

```
vcrypt.soap.auth=false
```

If SOAP Authentication is enabled, set the following properties:

```
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
```

For instructions on setting properties using the OAAM Admin Console, see *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*.

5. If you are installing the OAAM sample application on the same WebLogic Server domain where OAAM Server is running, then comment out the properties named `bharosa.cipher.encryption.algorithm.enum` that are related to encryption keys.

   If you are deploying on a non-WebLogic Server or a non-Identity Access Management WebLogic Server domain, then you must create the keystores `system_db.keystore` and `system_config.keystore` and set the values for the following properties:

   ```
   bharosa.cipher.encryption.algorithm.enum.DESede_config.keystorePassword
   bharosa.cipher.encryption.algorithm.enum.DESede_config.aliasPassword
   ```

6. Update the OAAM sample application so it picks up the changes made to the `oaam_custom.properties` file. Navigate to the Oracle WebLogic Administration Console and select **Deployments** and then **Summary of Deployments**. Click **Next** to the OAAM sample application and click the **Update** button. Click **Finish**.
7. Start the managed server.
8. Make changes to OAAM Web services security to allow access to OAAM SOAP services. By default they are protected by Oracle Web Services Manager (OWSM). The steps are as follows:
   a. Log in to Oracle Enterprise Manager Fusion Middleware Control of the Identity Management domain as a WebLogic Administrator using the URL:

      `http://`*`weblogic-admin-hostname`*`:`*`port`*`/em`

   b. Locate **oaam_server_server1** in the left pane by expanding **WebLogic Domain** and the OAAM domain under it.
   c. Right-click **oaam_server_server1** and select the **Web Services** menu option.
   d. Click **Attach Policies**.
   e. Select all the rows related to OAAM Web services in the next page and click the **Next** button.
   f. To enable SOAP authentication select the row **oracle/wss_http_token_service_policy** and click the **Next** button.

      To disable SOAP authentication, select the row **oracle/no_authentication_service_policy** and **oracle/no_authorization_service_policy** and click the **Next** button.

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

h. Restart the OAAM Server if required.

9. Navigate to the Oracle WebLogic Administration Console. Click **Lock and Edit** and select the **Deployments** node. On the Summary of Deployments page, find and select the OAAM sample application. Click **Start** > **Servicing all requests**. Click **Yes** to confirm.

10. Log in to the OAAM Admin application and import the OAAM snapshot `oaam_base_snapshot.zip` file into the system using the Oracle Adaptive Access Manager Administration Console. The snapshot contains policies, challenge questions, dependent components, and configurations that Oracle Adaptive Access Manager requires.

    For instructions on importing the OAAM snapshot, refer to *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*.

11. Navigate to the URL:

    `http://host-name:port/oaam_sample`

    You are shown the login page of the OAAM sample application.

12. Enter the username and then password in the next page. You be taken through the registration process.

    Note: The password must be `test` for the initial log in. You must change the password immediately.

2.2.3 Setting Up the Native In-Proc-Based OAAM Sample Application

This section contains instructions to set up the Native In-Proc-based OAAM sample application.

2.2.3.1 Pre-requisites

Before you can set up the Native In-Proc-based OAAM sample application you need:

- OAAM Admin to be installed, configured, and running
- Oracle Adaptive Access Manager SOAP service to be enabled and accessible from the host where the OAAM sample application is being deployed
- Details about the database host, username, and password used by Oracle Adaptive Access Manager
- The most recent OAAM Sample Application that illustrates Java API integration can be downloaded from My Oracle Support document ID 1351899.1.

2.2.3.2 Install and Configure

To set up the Native In-Proc-based OAAM sample application:

1. Create the `oaam_sample` directory.
2. Extract the `oaam_sample_inproc.zip` file into the `oaam_sample` directory.
3. Start the WebLogic Server.
4. Navigate to the Oracle WebLogic Administration Console at

   `http://oaam_host:port/console`

5. Deploy the OAAM Shared Library `$MW_HOME\Oracle_IDM1\oaam\oaam_libs\war\oaam_native_lib.war` as a shared library.
   a. Click **Deployments** under **IAMDomain** (in the left pane) in **Summary of Deployments** under the **Control** tab.
   b. Click the **Install** button. In the path specify `$MW_HOME\Oracle_IDM1\oaam\oaam_libs\war` and select **oaam_native_lib.war**. Click **Next**.
   c. Select the **Install this deployment as a library** option. Click **Next**.
   d. In the Select Deployments targets page, select the managed server from the list of servers and click **Next**. The name of the shared library is `oracle.oaam.libs`.

      If the managed server is OAAM Server then you do not need to create an OAAM data source. Otherwise create a data source with the JNDI name as `jdbc/OAAM_SERVER_DB_DS` and provide the connection details for the OAAM database schema.
   e. Click **Finish**.
6. Deploy the OAAM sample application as an application onto the same managed server where the OAAM Shared Library is deployed.
   a. Click **Deployments** under **IAMDomain** (in the left pane) in **Summary of Deployments** under the **Control** tab.
   b. Click the **Install** button. In the path, specify the location of the OAAM sample application. Click **Next**.
   c. Select the **Install this deployment as an application** option. Click **Next**.
   d. In the Select Deployments targets page, select the managed server from the list of servers and click **Next**.
   e. Click **Finish**.
7. Click **Activate Changes** under the **Change Center**.
8. In the deployment descriptor file, set the reference to the OAAM shared library `oracle.oaam.libs`.

   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>
   ```

   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>
   ```
9. Start the managed server.
10. Navigate to the Oracle WebLogic Administration Console. Click **Lock and Edit** and select the **Deployments** node. On the Summary of Deployments page, find and select the OAAM sample application. Click **Start** > **Servicing all requests**. Click **Yes** to confirm.

11. Log in to the OAAM Admin application and import the OAAM snapshot `oaam_base_snapshot.zip` file into the system using the Oracle Adaptive Access Manager Administration Console. The snapshot contains policies, challenge questions, dependent components, and configurations that Oracle Adaptive Access Manager requires.

    For instructions on importing the OAAM snapshot, refer to *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*.

12. Navigate to the URL:

    `http://managed_server:port/oaam_sample`

    You are shown the login page of the OAAM sample application.

13. Enter the user name and then password in the next page. You are taken through registration.

    Note: The password must be `test` for the initial log in. You must change the password immediately.

2.3 Integrating Virtual Authentication Devices, Knowledge-Based Authentication, and One-Time Password

An authentication flow is the process used to verify the identity of a person or other entity requesting access to a resource under security constraints. Multifactor authentication is a system wherein different factors are used in conjunction to authenticate the person or other entity.

The following integration flow example consolidates virtual authentication devices, knowledge-based authentication, and one-time password.

Virtual authentication devices are authenticator interfaces used to protect end-users during the process of entering and transmitting authentication credentials and provide them with verification that they are authenticating on the valid application. OAAM includes a suite of highly secure virtual authentication devices as samples to deploy if you choose to. Alteration of these samples is considered custom development. If the deployment supports localization, globalized virtual authentication device image files including registration flows must be developed by the deployment team.

Knowledge-based authentication (KBA) is a secondary authentication method that provides an infrastructure based on registered challenge questions. It enables end-users to select questions and provide answers which the system uses later on to challenge them when necessary. Security administration include:

- Registration logic to manage the registration of challenge questions and answers
- Answer Logic to intelligently detect the correct answers in the challenge response process
- Validations for answers given by a user at the time of registration

For information, see "Managing Knowledge-Based Authentication" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*.

One-time Password is a risk-based challenge solution consisting of a server generated one time password delivered to an end-user through a configured out-of-band channel. Supported OTP delivery channels include short message service (SMS), email, and instant messaging IM). You can use OTP to compliment KBA challenge or instead of KBA.

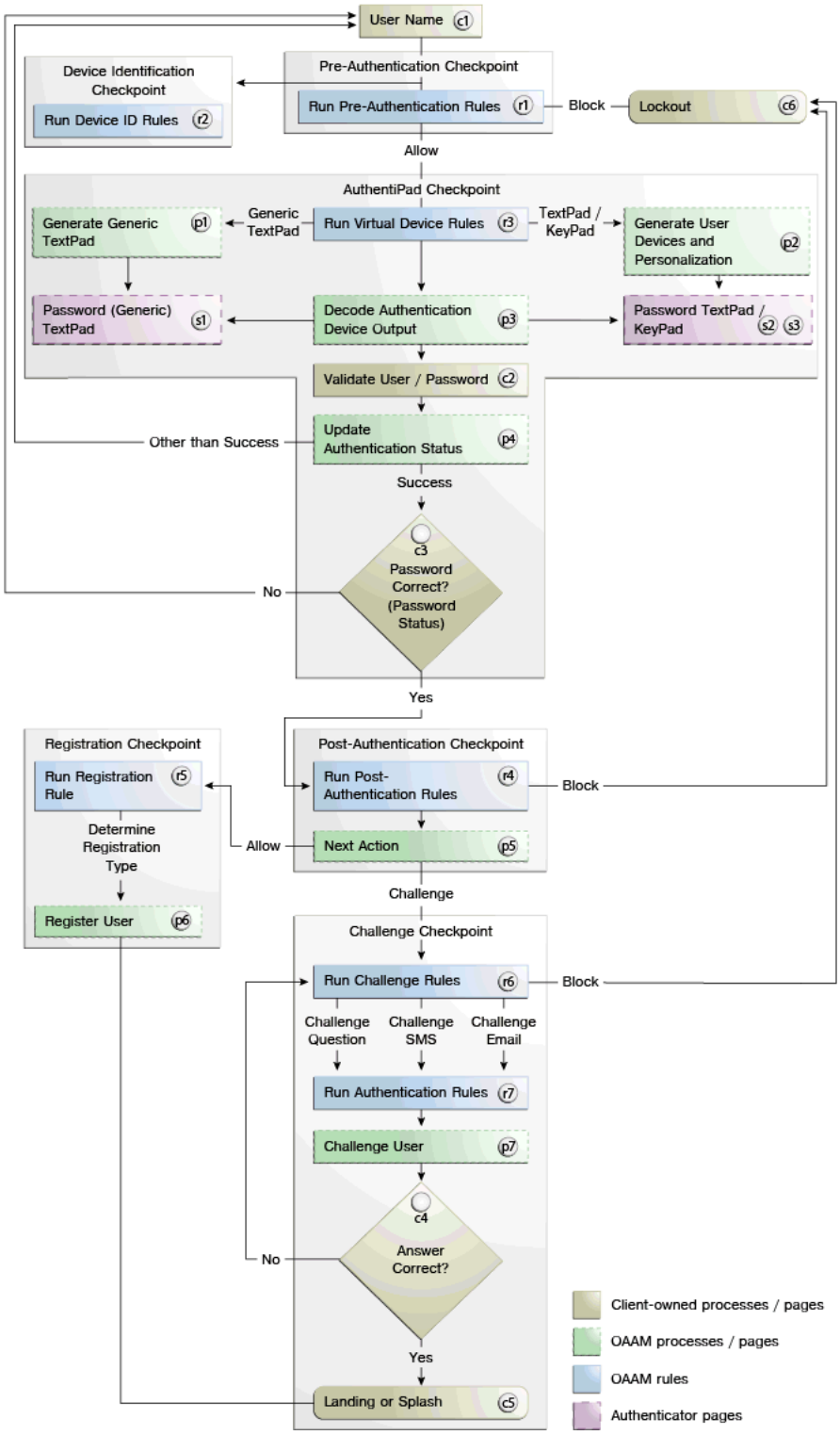
For information, see "Setting Up OTP Anywhere" in *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*.

Figure 2–3 illustrates an authentication flow example that uses these three solutions (virtual authentication devices, KBA, and OTP). The flow illustrated is an example and that other authentication flows are possible.

Details about the checkpoints and rules are explained in the following sections:

- User Name Page (c1)
- Device Fingerprint Flow (r2)
- Run Pre-Authentication Rules (r1)
- Run Virtual Authentication Device Rules (r3)
- Decode Virtual Authentication Device Input (p3)
- Validate User and Password (c2)
- Run Post-Authentication Rules (r4)
- Check Registration for User (p5)
- Run Registration Required Rules (r5)
- Enter Registration Flow (p6)
- Run Challenge Rules (r6)
- Run Authentication Rules (r7)
- Challenge the User (p7)
- Check Answers to Challenge (c4)
- Lock Out Page (c6)
- Landing or Splash Page (c5)

Figure 2–3 Virtual Authentication Devices, Knowledge-Based Authentication, and OTP Scenario



2.3.1 User Name Page (c1)

When the application uses a custom login page, the login page must be split into two pages. The user is presented with a page in which he is asked to submit his user name: The user enters the login ID (user name) in the first page, and this user name is stored in the HTTP session. The user name page is followed by a transient page to capture the Flash and secure cookies and for fingerprinting the user device. Figure 2–4 shows an example of the user name page.

Figure 2–4 User Name Page

If OAAM is configured to use a single login page, virtual authentication devices are not used in any flow. The OAAM AuthenticationPad Policy is configured to use the OAAM HTML Pad for Challenge SMS, Registered Image and Caption, Challenge Email, and Challenge Question rules or OAAM Server is configured to use the OAAM HTML Pad using the OAAM Extensions Shared Library. For details on the Single Login Page, see Section 8.5, "Enabling the Single Login Page."

2.3.2 Device Fingerprint Flow (r2)

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 the user uses it to log in to the system. The fingerprinting process produces a fingerprint that is unique to the user. The fingerprint details help in identifying a device, check whether it is secure, and determine the risk level for the authentication or transaction.

Table 2–1 lists the APIs used for device fingerprinting.

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 4.5.33, "updateLog."
Oracle Adaptive Access Manager Sample	`handleJump.jsp`	`handleJump.jsp`: 1. Sets the client's time zone 2. Sets a secure cookie 3. Sets the browser fingerprint 4. Sets the status to pending 5. Calls the pre-authentication rules; expects `ALLOW` to allow the user to proceed or `BLOCK` or `ERROR` to stop the user from continuing 6. Stores the session data `bharosaSession` 7. Forwards the user to the password page `password.jps`
Oracle Adaptive Access Manager Sample	`handleFlash.jsp`	`handleFlash.jsp` sets the Flash cookie 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 sample code shows how to fingerprint the device using browser and Flash cookies. See the 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);
```

2.3.3 Run Pre-Authentication Rules (r1)

Additionally, 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
- `BLOCK` to block the user from proceeding

Table 2–2 lists the APIs used for pre-authentication.

Table 2–2 Pre-Authentication Rules Reference APIs

Module	APIs	Description
Server	`VCryptRulesEngine::processRules()`	For method details, see Section 4.5.25, "processRules."
Oracle Adaptive Access Manager Sample	`handleJump.jsp`	`handleJump.jsp`: 1. Invokes the pre-authentication rules 2. Returns `ALLOW` to proceed to the password page `password.jsp` or `BLOCK` or `ERROR` to signal an error 3. Stores `bharosaSession`
BharosaHelper	`BharosaHelper::runPreAuthRules()`	

2.3.4 Run Virtual Authentication Device Rules (r3)

The Authentipad checkpoint determines the virtual authentication device to use. If the user has not registered image and phrase, the rule assigns the Generic TextPad; otherwise, if the user has registered, the rule assigns 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 personalized Textpad is a device for entering a password or PIN using a keyboard. This method of data entry helps to defend against phishing. TextPad can be deployed as the default for all users in a large deployment then each user individually can upgrade to another device if they want. The personalized image and phrase a user registers and sees every time they log in to the valid website serves as a shared secret between user and server.

The Keypad is a device for entering passwords, credit card number, and so on, using a keyboard for entry. The KeyPad protects against Trojan or keylogging.

Table 2–3 lists the APIs used to run virtual authentication device rules.

Table 2–3 Virtual Authentication Device Rules APIs

Module	APIs	Description
Server	`VCryptRulesEngine::processRules()`	For method details, see Section 4.5.25, "processRules."
Oracle Adaptive Access Manager Sample	`password.jsp`	`password.jsp`: 1. Invokes rules to identify the user's virtual authentication device type 2. Creates the virtual authentication device, names it, and sets all initial background frames 3. Invokes `kbimage.jsp` as configured 4. Forwards to page `handlePassword.jsp`
BharosaHelper	`BharosaHelper::getAuthentiPad()`	

2.3.5 Generate a Generic TextPad (p1)

If the user has not yet registered for image and phrase with OAAM, he is shown a generic, non-personalized TextPad, as seen in Figure 2–5.

There is no specific image or phrase on this device, but there is a timestamp to help prove the legitimacy of the login prompt.

Figure 2–5 Generic, Non-Personalized TextPad

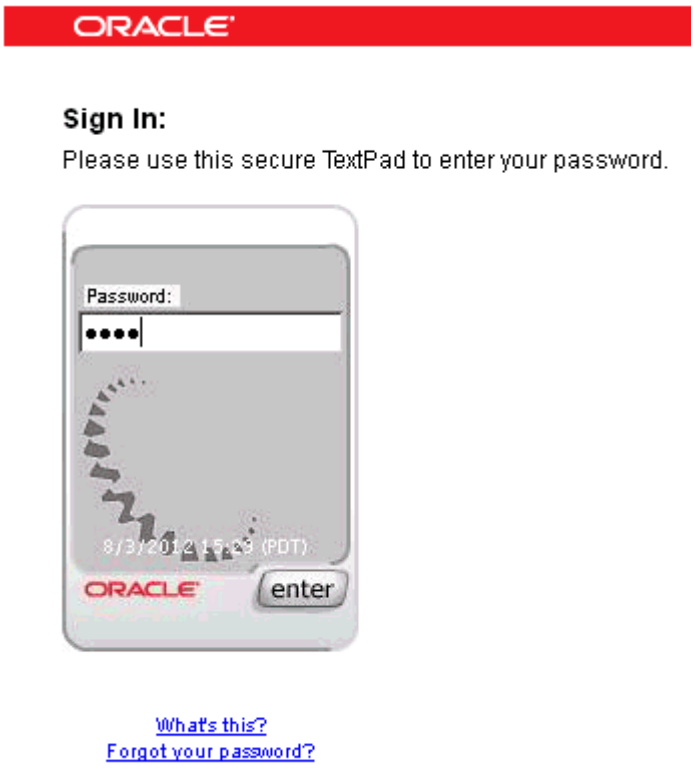


Table 2–4 lists the APIs used to generate a generic TextPad.

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 4.5.19, "getUserByLoginId."
Oracle Adaptive Access Manager Sample	`Password.jsp`	`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()`	

2.3.6 Generate a Personalized TextPad or KeyPad (p2)

If the user has registered a phrase and image with OAAM, a personalized TextPad is used for the user. Figure 2–6 andFigure 2–7 illustrate personalized text and key virtual authentication devices to be generated.

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 4.5.19, "getUserByLoginId."
Oracle Adaptive Access Manager Sample	`password.jsp`	`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()`	

2.3.7 Display TextPad and KeyPad (s2 and s3)

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`	`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 devices

BharosaHelper	`BharosaHelper:: createPersonalizedAuthentiPad ()`
	`BharosaHelper::createAuthentiPad()`
	`BharosaHelper::imageToStream()`	

Client	`AuthentiPad::getHTML()` `KeyPadUtil::encryptImageToStream()`	

Figure 2–6 shows a personalized textpad.

Figure 2–6 Personalized TextPad

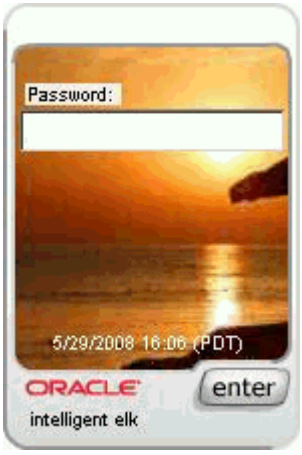
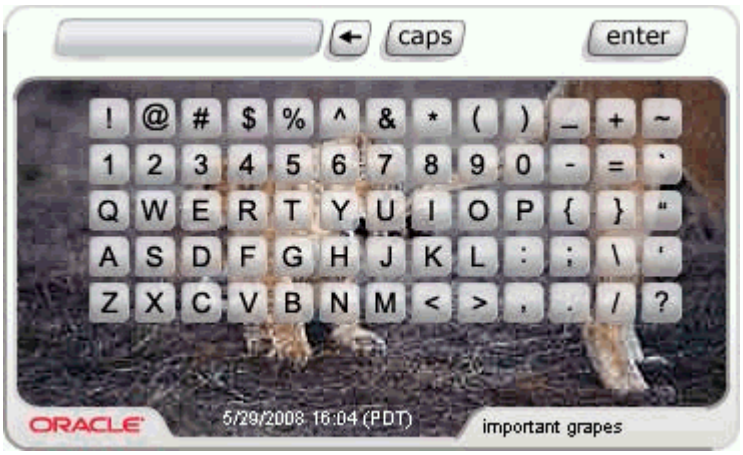


Figure 2–7 shows a personalized keypad.

Figure 2–7 Personalized KeyPad



Note the image and phrase are embedded in the devices along with a current timestamp in the user's local time zone.

2.3.8 Decode Virtual Authentication Device Input (p3)

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 the system uses it 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`	`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`	

2.3.9 Validate User and Password (c2)

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	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

2.3.10 Update Authentication Status (p4)

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 4.5.32, "updateAuthStatus."
Oracle Adaptive Access Manager Sample	handlePassword.jsp	handlePassword.jsp ■ Retrieves the password ■ Decodes the password ■ Validates the user ■ Forwards to registerImageandPhrase, or challenges a registered user
BharosaHelper	BharosaHelper::updateStatus()	

2.3.11 Password Status (c3)

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

2.3.12 Run Post-Authentication Rules (r4)

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

- **ALLOW**
- **BLOCK**
- **CHALLENGE**

If the outcome of Post-Authentication is ALLOW, then OAAM determines if the user has to be registered and sends the user to the Registration flow. If the outcome of the Post-Authentication is BLOCK, then the user is blocked and will not be able to access the web application that the user tried to access. If the outcome of Post-Authentication is

CHALLENGE and if the user is already registered for at least one of the challenge mechanisms, OAAM sends the user to the Challenge flow.

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 4.5.25, "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()	

2.3.13 Check Registration for User (p5)

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

2.3.14 Run Registration Required Rules (r5)

Business and security requirements specify whether 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 4.5.25, "processRules."
Oracle Adaptive Access Manager Sample	`password.jsp`	`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()`	

2.3.15 Enter Registration Flow (p6)

Registration is the enrollment process, the opening of a new account, or other event where information is obtained from the user. The Registration flow allows the user to register for questions, image, phrase, and OTP (email, phone, and so on). Once the user is successfully registered, you can use KBA and OTP as secondary authentication factors to challenge the user.

Table 2–12 describes the modules and APIs in the Registration flow.

Table 2–12 Registration Flow

Module	APIs	Description
Server	`VCryptRulesEngine::processRules()`	For method details, see Section 4.5.25, "processRules."
Oracle Adaptive Access Manager Sample	`registerImagePhrase.jsp`	`registerImagePhrase.jsp` ■ Assigns new image and caption to user ■ Assigns new image and caption to user ■ Forwards to page `handleRegisterImagePhrase.jsp`

Table 2–12 (Cont.) Registration Flow

Module	APIs	Description
	registerQuestions.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	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	

2.3.16 Run Challenge Rules (r6)

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 through SMS, to challenge user with OTP
- **ChallengeEmail** to challenge user with OTP through 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 4.5.25, "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()	

2.3.17 Run Authentication Rules (r7)

`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 assigns the QuestionPad. If the user is to be challenge with an OTP, the rule assigns the TextPad.

2.3.18 Challenge the User (p7)

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 through 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`, and so on) 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 2.3.4, "Run Virtual Authentication Device Rules (r3).". 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()`	

2.3.19 Check Answers to Challenge (c4)

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 4.5.25, "processRules," and Section 4.5.32, "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`

2.3.20 Lock Out Page (c6)

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

2.3.21 Landing or Splash Page (c5)

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

3

Integrating Native .NET Applications

ASP.NET applications can integrate with Oracle Adaptive Access Manager using the .NET API provided by Oracle Adaptive Access Manager to add various OAAM feature, such as, virtual authentication devices, and KBA, and the OAAM Risk Engine.

This chapter provides details on how to use the OAAM .NET API for integrating ASP .NET applications. Descriptions are also provided on the OAAM sample applications, which 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 websites, Web applications, and Web services. OAAM provides an OAAM .NET development kit (SDK). The OAAM .NET SDK is used for integrating ASP.NET applications with OAAM. It includes the OAAM .NET APIs that are exposed by the OAAM .NET library, OAAM sample .NET applications, OAAM flash movie page (which OAAM uses to collect the fingerprint in device identification), and other files that are required for .NET native integration. ASP.NET applications, written in any ASP.NET language, can use the OAAM .NET API to call Oracle Adaptive Access Manager.

The OAAM .NET API communicates with the OAAM server using Simple Object Access Protocol (SOAP). SOAP is a protocol specification for exchanging structured information in the implementation of Web Services in computer networks.

3.2 Oracle Adaptive Access Manager .NET SDK

The .NET-based OAAM Sample application that illustrates .NET API integration can be downloaded from My Oracle Support document ID 1627136.1.

OAAM .NET SDK is packaged in `oaam_native_dot_net.zip` in `$ORACLE_HOME/oaam/oaam_libs/dotNet/`.

The sample .NET applications that enable OAAM features require the integration of the OAAM .NET APIs found in the SDK package. The developer must extract the content of the archive to the root directory of the web application.

The OAAM .NET libraries are located in the `/bin` directory in the extracted SDK package.

3.3 Configuration Properties

The Oracle Adaptive Access Manager .NET SDK includes property files that specify values for the configuration used by the OAAM .NET 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 OAAM .NET API uses properties to read configurable values at run time, such as the location of images for virtual authentication devices. Virtual authentication devices are controls for user input and provide a 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 the OAAM .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/
<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 OAAM .NET SDK. `BharosaUtils.exe` can be found at the `/bin` directory after extracting OAAM .NET SDK package.

An encryption key (arbitrarily selected by the user) is required to encrypt and decrypt values. This key is available to the OAAM .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 outputted 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-encrypt
Enter text to be encrypted again: string-to-encrypt
vCCKC19d14a39hQSKSirXSiWfgbaVG5SKIg==
```

3.3.3 Using User-Defined Enumerations to Define Elements

Visual Studio 2005 enables you to use enumerations defined in the .NET Framework. 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

An example of a user-defined enumeration is presented below.

```
#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
```

An example of the use of the previous user-defined enumeration in application code is shown as follows:

```
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 you can use OAAM APIs to support common OAAM scenarios. You can also refer to the OAAM 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 run time.

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);
//createUser API calls OAAM Server to create a user in database. New user will be
//returned.
```

```
    // 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 more details, see the OAAM sample applications listed in Section 3.5.2, "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.

Some APIs are:

- `handletrackerRequest()`: creates the signatures required to fingerprint the device
- `updateLog()`: Updates the user node log and if required, creates the CookieSet also.
- `createTransaction()`: creates a data entry for the transaction at OAAM Server
- `updateTransaction()`: updates a given transaction
- `updateTransactionStatus()`: updates the status of a transaction in OAAM Server
- `markDeviceSafe()`: marks the device to be safe when needed
- `isDeviceMarkedSafe()`: checks whether the device has been marked safe

The following code sample illustrates the use of this updateLog() 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);
}
```

By calling the `updateLog()` API, user information, with browser/flash fingerprint information, will be sent to the OAAM Server through a SOAP call. OAAM Server will return a new fingerprint cookie if fingerprint information being sent matches the values stored at the OAAM Server. If the user information has not been obtained, OAAM uses the `handleTrackerRequest()` API to collect device information as used in the OAAM Sample .NET Application.

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 *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

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

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

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

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

3.4.4 Validate 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 code example 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 OAAM sample application in Integration Example Using the Sample
// Applications
// for details on displaying the questions in the UI and processing the user input
// 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 Integration Example Using the Sample Applications;
// 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 OAAM sample applications listed in Section 3.5.2, "ASP.NET Applications."

3.4.5 Reset Challenge Failure Counters

Oracle Adaptive Access Manager records the number of wrong answers to the questions posed to the user in the failure counters. Oracle Adaptive Access Manager uses failure counters 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 (for example, `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 Specify 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 Trace 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>
```

3.4.9 .Net API Support for X.509 SSL Certificate Configuration

The .Net API supports X.509 SSL certificate configuration when using SOAP to call the OAAM server. If you have an OAAM server deployed in an environment that requires an X.509 certificate in the SSL authentication process(or 2-way SSL), use the following APIs to add/remove the certificate to the OAAM .NET SOAP client.

For .NET 2.0, the APIs in the `Bharosa.VCrypt.ClientIBharosaProxy` interface are:

- void AddClientCertificate()
- void AddClientCertificate(X509Certificate clientCert)
- void AddClientCertificate(string certFilePath, string password)
- void RemoveClientCertificate(X509Certificate clientCert)

The properties to enable adding and removing of the X.509 SSL certificates when making SOAP calls to the OAAM Server when two-way SSL is required are documented below. You must set the SOAP user/password and group according to the configuration.

```
<add key="BharosaSOAPURL"
value="https:{OAAM SOAP SERVICE URL}"/>
<add key="BharosaSOAPUser"     value="ruleAdmin1"/>
<add key="BharosaSOAPPassword" value="welcome1"/>
<add key="BharosaSOAPDomain"   value="OAAM_Webservices_Group"/>
<add key="BharosaSOAPTrustAllServerCert"   value="true"/>
<add key="BharosaSOAPClientCertFilePath"
value="path/to/the/client/certificate/file"/>
<add key="BharosaSOAPClientCertFilePassword"
value="{password to open client certificate file}"/>
```

Use '`<add key="BharosaSOAPTrustAllServerCert" value="true"/>`' only for debug/development purpose to let the .NET SOAP client accept all server-side SSL certificate during the SSL authentication process. Do not use the property in a production environment.

3.5 OAAM Sample Applications as Reference for Integration

OAAM sample applications are provided in the SDK as references to illustrate how to integrate an application. This section provides details on the contents and flow each OAAM application demonstrates.

3.5.1 Downloading the Sample Package

The .NET-based OAAM Sample application that illustrates .NET API integration can be downloaded from My Oracle Support document ID 1627136.1.

The OAAM Sample applications are for demonstration purposes to familiarize you with OAAM APIs. They are not intended to be used as production code since they only provide basic elements of API usage. If you are implementing a .NET integration, you can develop your application using the OAAM Sample applications as reference. Custom applications developed for these deployments are not supported directly; however, Oracle Support Services can assist you with product issues, such as if you were to encounter problems when using the provided APIs.

3.5.2 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, virtual authentication devices, and KBA.
`SampleWebAppTracker`	This application demonstrates integration of Oracle Adaptive Access Manager Risk Engine to SampleWebApp.
`SampleWebAppAuthTracker`	This application demonstrates integration of Oracle Adaptive Access Manager Risk Engine and virtual authentication device to SampleWebApp.
`SampleKBATracker`	This application demonstrates integration of the Oracle Adaptive Access Manager Risk Engine and KBA to SampleWebApp.

3.5.3 OAAM Sample Application Details

Details about the four applications are provided in this section.

3.5.3.1 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

3.5.3.2 SampleWebAppTracker

This application contains the following pages that demonstrate integration of the Oracle Adaptive Access Manager Risk Engine to the OAAM sample application listed prior. The Oracle Adaptive Access Manager Risk Engine helps the OAAM server collect multiple kinds of user information including the User ID entered, device fingerprint collected by the OAAM embedded flash movie, IP information, and so on. The integrated web application could call appropriate SOAP APIs at the required checkpoint. OAAM Server will run pre-defined authentication rules on collected information according to authentication rules defined through the OAAM Admin console. The authentication result will be returned so that the protected web application could take corresponding actions accordingly.

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 must 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 fingerprint of the user device
2. `LoginJumpPage.aspx`
   - Loads the user from OAAM by calling `AppUtil.InitUser()` (AppUtil is included in the SDK package). If the user is not found, a new user record will be created. By calling `BharosaClientFactory.getProxyInstance()`, OAAM gets a reference to the `IBharosaProxy` interface. This interface exposes the multiple OAAM .NET SOAP APIs for integrating .NET applications. APIs call in `AppUtil.InitUser()`: `getUserByLoginId()`, `getUser()`, `createUser()`, `setUserStatus()`, `getUserStatus()`, and `setPin()`.
   - 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 fingerprint details. `CookieManager.aspx` records the fingerprint in OAAM 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 OAAM), redirects the browser to `LoginHandlerPage.aspx`
3. `LoginHandlerPage.aspx`
   - Records the user login attempt with OAAM by calling `AppUtil.InitTracker()`
   - Validates the login and password information
   - Updates OAAM with the password validation status (success/wrong user/wrong password/disabled user, and so on) 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 OAAM 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

3.5.3.3 SampleWebAppAuthTracker

This application contains the following pages that demonstrate integration of Oracle Adaptive Access Manager Risk Engine and a virtual authentication device to the

OAAM sample application listed prior. This application collects the password using authenticators offered by OAAM.

Authenticator functionality refers to the use of the OAAM Virtual Authentication Device used to collect credentials. By calling the OAAM .NET API to run the pre-authentication rule, a user might be blocked before he can see the OAAM virtual authentication device. By running the Authentipad rule, the OAAM Virtual Authentication Device will be selected/created for the user and rendered on the password page.

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 must 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 fingerprint of the user device
2. `LoginJumpPage`.aspx
   - Loads the user from OAAM 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 fingerprint details. `CookieManager.aspx` records the fingerprint in OAAM 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 OAAM), redirects the browser to `LoginHandlerPage.aspx`
3. `LoginHandlerPage.aspx`
   - Records the user login attempt with OAAM 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 OAAM

   b. Runs pre-authentication rules by calling the `AppUtil.RunPreAuthRules()`

   c. If the pre-authentication rules return `block`, blocks the user login after updating OAAM 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 OAAM with the password validation status (`success/wrong user/wrong password/disabled user`, and others) 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 OAAM 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

3.5.3.4 SampleKBATracker

This application contains the following pages that demonstrate integration of OAAM authenticator, risk engine, and KBA and KBA (Knowledge Based Authentication) to the OAAM sample application listed prior. This application shows authentication mechanisms using password and KBA authenticators offered by OAAM. OAAM KBA enables the ability to let the user register challenge questions and challenge the user at some point. For example, based upon the result of post-authentication rules, the integrated web application could decide to challenge a user using registered question/answer pairs.

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 must 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 fingerprint of the user device
2. `LoginJumpPage.aspx`
   - Loads the user from OAAM 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 fingerprint details. `CookieManager.aspx` records the fingerprint in OAAM 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 OAAM), redirects the browser to `LoginHandlerPage.aspx`

3. `LoginHandlerPage.aspx`
   - Records the user login attempt with OAAM 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 OAAM

   b. Runs pre-authentication rules by calling the `AppUtil.RunPreAuthRules()`

   c. If the pre-authentication rules return `block`, blocks the user login after updating OAAM 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 the 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 OAAM with the password validation status (`success/wrong user/wrong password/disabled user`, and others) 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 depending on 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 Security Profile for the user
   - If the post authentication rule returns `RegistrationOptiona`l, 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 QuestionPad with one of the questions already registered by the user
   - The answer is validated by calling `proxy.authenticateQuestion()` and the result is updated in OAAM 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 reenter the answer or be redirected to the block page after updating the block status in OAAM
   - The number of attempts that a user gets to answer a question correctly is set by the rule administrator for OAAM
   - 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

3.5.4 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 Internet Information Server (IIS) 6.0 on Windows Server 2003. You can use `SampleWebApps` to load and view all applications together using Visual Studio.

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

3.5.4.1 Modifying the web.config File

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

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

3.5.4.2 Setting Properties for Images

For OAAM sample applications integration with OAAM 11g, set `bharosa.image.dirlist` in `bharosa_app.properties` to the path where `oaam_images` directory could be found. The `oaam_images` directory is located at: `${ORACLE_HOME}/oaam/`. The `oaam_images` directory includes images that OAAM will use to generate a virtual authentication device.

The directory 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 directory named `oaam_images` and this directory is under the `root` directory of the web application, the path should be: `${Application_HOME}/oaam_images/`

Make sure `lookup.properties` is created/contained in the `/bharosa_properties/` directory, which should list all the properties files that need to be read. It can be obtained from:

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

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

3.5.4.3 Running the Application

For developers who have access to Microsoft Visual Studio 2005 to test the web applications, build the solution after making all the prior changes and debug it.

For deployment of these applications, follow these guidelines:

- The system should be Windows Server 2003
- The application server should be installed using **Control Panel** > **Add or Remove Programs** > **Add/Remove Windows Components**. Microsoft Internet Information Server (IIS) and ASP.NET should be enabled
- Create a new website using Internet Information Services (IIS) Manager by running `inetmgr` in the command window
- Ensure that the ASP.NET version is set to version 2.0 through the **ASP.NET** tab in the website's properties;
- Ensure that ASP.NET version 2.0 is set to `allowed` in Internet Information Services (IIS) Manager. If there is no ASP.NET version 2.0 extension, add a new web service extension manually. Go to `C:\WINDOWS\Microsoft.NET\Framework`, there should be a directory named `v2.0.50727` or similar if ASP.NET version 2.0 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**, select **Local System** on the right of the **Predefined** option if there is a problem accessing `C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\Temporary ASP.NET` Files when opening web application pages.

3.5.5 Example: Enable Transaction Logging and Rule Processing

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

Prerequisites:

- Transaction definitions in `Sample_Transaction_Defs.zip` need to be available in Oracle Admin. Import the transaction definitions using OAAM Admin.
- Transaction policies defined in models.zip should be available in OAAM Admin
- Following properties must exist in `bharosa_app.properties` at the OAAM Admin and the .NET client side:

  Enumeration for Transaction Status

```
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
```

  Enumeration for Checkpoints

```
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
```

Admin Options for the 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.

3.5.6 OAAM .NET API

For more information on the APIs listed in this section, see *Oracle Fusion Middleware Java API Reference for Oracle Adaptive Access Manager*.

Note: `isElementInList()`, `getListElements()` and `updateList()` APIs do not support update and other actions in the alert group lists.

Table 3–3 describes the .NET APIs available in OAAM.

Note: The generateOTP() API has been deprecated in the OAAM JAVA and SOAP APIs. Please use the getOTPCode() API instead when writing your production code. For details on how to use the getOTPCode() API, see the *Oracle Fusion Middleware Java API Reference for Oracle Adaptive Access Manager*.

Table 3–3 OAAM .NET API

API	Description
`handleTrackerRequest()`	`handleTrackerRequest()` creates the signatures required to fingerprint the device. This method takes the requestTime as input.
`handleTransactionLog()`	`handleTransactionLog()` creates the signatures required to fingerprint the device.
`createTransactions()`	`createTransactions()` creates an OAAM Transactions in bulk Return response object for each create request.
`updateTransactions()`	`updateTransactions()` updates Transactions in bulk. If there are errors in any update, it will proceed with the next transaction and return a response for each request.
`updateLog()`	`updateLog()` updates the user node log, and if required, it creates the CookieSet also.
`updateTransactionStatus()`	`updateTransactionStatus()` updates the transaction's status.
`updateAuthStatus()`	`updateAuthStatus()` updates the authentication status for the request. All parameters must have valid values for this function to work correctly.
`markDeviceSafe()`	`markDeviceSafe()`: marks the device as safe.
`getUser()`	`getUser()` returns the user details without the password and PIN for the customer and group. If the user is not valid, then all the values in the object is null. If there are any unexpected errors, Null is returned.

Table 3–3 (Cont.) OAAM .NET API

API	Description
`getUserByLoginId()`	`getUserByLoginId()` returns the user details without the password and PIN for the customer and group. If the user is not valid, then all the values in the object is null. If any unexpected errors occur, Null is returned.
`createUser()`	`createUser()` creates a user in the authentication database. It returns null if user is null.
`setUser()`	`setUser()` updates the user in the authentication database. It returns null if the user is null or the `customerId` attribute in user is invalid.
`setPin()`	`setPin()` sets a new PIN for the user. It returns whether the operation was Success or Failure.
`setPassword()`	`setPassword()` sets a new password for the user. It returns whether the operation was `Success` or `Failure`.
`setCaption()`	`setCaption()` sets a new caption for the user. If the caption is null, a caption with the default locale is set.
`setImageAndCaption()`	`setImageAndCaption()` sets the image and caption for the user. If the caption is null, a caption with the default locale and default text is set.
`setUserAuthMode()`	`setUserAuthMode()` sets an authentication mode for the user. The Authentication mode can be full keypad, and so on.
`setGroupUserAuthMode()`	`setGroupUserAuthMode()` uses batch updates to set authentication mode. A failure is not guaranteed to leave the system in the old authentication mode since failure may occur in a later batch after initial batches are saved.
`getImage()`	`getImage()` gets the image path for the user.
`getCaption()`	`getCaption()` gets the caption for the user.
`getImageAndCaption()`	`getImageAndCaption()` gets the image path and caption for the user.
`getLocalizedCaption()`	`getLocalizedCaption()` gets the localized caption for the user.
`getUserAuthMode()`	`getUserAuthMode()` gets the authentication mode for the user.
`getUserStatus()`	`getUserStatus()` gets the status for the user.
`getSignOnQuestions()`	`getSignOnQuestions()` gets all the secret questions available for the user. It returns the 2-D array object containing the questions to ask. First dimension denotes the number of (configurable) question pick sets to display to the user and the second dimension denotes the number of questions in each pick set.
`getAllMappedSignOnQuestions()`	`getAllMappedSignOnQuestions()` gets user questions
`setUser()`	`setUser()` updates the user in the authentication database and returns null if user is null or the `customerId` attribute for the user is invalid.
`addQuestions()`	`addQuestions()` add questions to the customer. It expects the number of questions to be exactly equal to the required number of questions. Calling this method will delete any previously existing questions. Success indicates adding all questions; failure means none of the questions are added.
`deleteQuestion()`	`deleteQuestion()` deletes the question for the specified user
`getSecretQuestion()`	`getSecretQuestion()` gets a secret question for the user. It returns the object containing the question to ask.
`moveToNextSecretQuestion()`	`moveToNextSecretQuestion()` moves the current secret question for the user to the next question. It returns the object containing the question to ask and null in case of errors.

Table 3–3 (Cont.) OAAM .NET API

API	Description
`authenticateQuestion()`	`authenticateQuestion()` authenticates the question/answer. It returns the description result of the authentication attempt.
`authenticateQuestionForCSR()`	`authenticateQuestionForCSR()` is the method to authenticate question/answer for customer care. It returns the description result of the authentication attempt.
`processRules()`	`processRules()` runs the rules and returns rules result. The attribute 'response' in `VCryptRulesResult` returns a `Success`. `VCryptResponse` with no session is set by default. The attribute `alertIdList` is null if the rules triggered have no corresponding alerts. The `transactionLogId` attribute is set if the property `vcrypt.tracker.rule.process.autoTransactionLog.disable` is set to false.
`createList()`	`createList()` creates a new list of the given list type.
`updateList()`	`updateList()` updates the given list with new elements. The list name must be an existing one. Duplicate and invalid elements in `elementsToAdd` are ignored. Non-existing and invalid elements in `elementsToRemove` are ignored. Update of alert group list is not supported.
`getLists()`	`getLists()` gets a list of groups when given a group type.
`isElementInList()`	`isElementInList()` checks whether the element given is in the list. Checks of elements in an alert group list are not supported.
`getFinalAuthStatus()`	`getFinalAuthStatus()` returns the final authentication status of a user given the user ID of the user. This method can only go back up to 30 days.
`setTemporaryAllow()`	`setTemporaryAllow()` sets a temporary allow for the user.
`getActionCount()`	`getActionCount()` gets the action count for the given `actionEnumId`. Consult your configuration for available action enums. The property `[rule.action.enum.<actionName>.incrementCacheCounter]` is to be set to `true` to increment the counter for the action corresponding to `<actionName>`. If it is not set or set to false, the method returns successfully, the value which is present in the cache, but this value may not reflect the exact action count.
`CancelAllTemporaryAllows()`	`CancelAllTemporaryAllows()` cancels all the unused temporary allows for the user.
`getRulesData()`	`getRulesData()` returns all the rules executed for the given session Id, and provides basic information of what rules were triggered. It does not provide complete hierarchy information. Rules execution data is persisted asynchronously and may not be available immediately.
`getRulesDataForLastSession()`	`getRulesDataForLastSession()` returns all the rules executed for the given `customerId` for the past session and provides basic information about what rules were triggered. It does not provide complete hierarchy information. Rules execution data is persisted asynchronously and may not be available immediately.
`resetUser()`	`resetUser()` resets all profiles set for the user. This includes registration, questions, images and phrases selected or assigned to the user
`processPatternAnalysis()`	`processPatternAnalysis()` triggers the pattern data processing for autolearning. This method does not perform any other activity other than autolearning pattern analysis.
`getList()`	`getList()` gets a list of groups given a group type

Table 3–3 (Cont.) OAAM .NET API

API	Description
`ClearSafeDeviceList()`	`ClearSafeDeviceList()` clears the safe device list of the user associated with this request
`resetChallengeFailureCounters()`	`resetChallengeFailureCounters()` resets challenge failure counters.
`generateOTP()`	The `generateOTP()` API has been deprecated in the OAAM JAVA and SOAP APIs. Please use the getOTPCode() API instead when writing your production code. For details on how to use the getOTPCode() API, see the *Oracle Fusion Middleware Java API Reference for Oracle Adaptive Access Manager*.
`getOTPCode()`	`getOTPCode()` can be called 'n' number of times to get the OTP code for the given request identifier. If there is no OTP code that exists for the given request identifier, then a new OTP code will be generated. If the OTP code exists for the given request identifier and `overwriteIfexists` is `true` then the new OTP code will be generated. If the OTP code exists and the OTP code is not expired, then the same OTP code will be returned by renewing the expiry, otherwise the new OTP code will be returned. The OTP code can be retrieved from the `VCryptObjectResponse` object as the following property `String otpCode = (String) vcryptResponseObj.getObject();`
`validateOTPCode()`	`validateOTPCode()` validates the OTP code for the given request identifier and challenge type. This method can be called 'n' number of times to validate the OTP code for a given request identifier. If the OTP code exists and has not expired and the given OTP code matches the existing OTP code, then it returns the response with `OTP_CODE_MATCHED` value. If the OTP code exists and has not expired and the given OTP Code does not match the existing OTP code, then it returns a response with `OTP_CODE_NOT_MATCHED` value. If the OTP code exists and expired then it returns a response with the `OTP_CODE_EXPIRED` value. If the OTP code does not exists, then it returns `OTP_CODE_DOESNOT_EXISTS`. The OTP validation result can be retrieved from the `VCryptObjectResponse` object as follows: `TrackerAPIUtil.OTPValidationResult otpCode = (TrackerAPIUtil.OTPValidationResult) vcryptResponseObj.getObject();`
`resetChallengeCounter()`	`resetChallengeCounter()` resets the challenge counter.
`incrementChallengeCounter()`	`incrementChallengeCounter()` increments the challenge counter.
`createOrUpdateEntities()`	`createOrUpdateEntities()` creates or updates entities.
`void AddClientCertificate()`	Reads "`BharosaSOAPClientCertFilePath`" and "`BharosaSOAPClientCertFilePasswod`" properties if set in `web.config` and gets the X.509 certificate and adds it to the SOAP clients. Supports X.509 SSL certificate configuration when using SOAP to call the OAAM server.
`void AddClientCertificate(X509Certificate clientCert)`	Adds the given X.509 certificate to the SOAP client. Supports X.509 SSL certificate configuration when using SOAP to call the OAAM server.
`void AddClientCertificate(string certFilePath, string password)`	Gets the certificate in the given file path using the given password and then adds the certificate to the SOAP client. Supports x.509 SSL certificate configuration when using SOAP to call the OAAM server.
`void RemoveClientCertificate(X509 Certificate clientCert)`	Removes the given certificate from the SOAP client if it is under the .NET framework Supports x.509 SSL certificate configuration when using SOAP to call the OAAM server.

4

Natively Integrating OAAM with Java Applications

You can 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 chapter contains the following sections:

The most recent OAAM Sample Application that illustrates Java API integration can be downloaded from My Oracle Support document ID 1351899.1.

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 an Oracle WebLogic Server instance where the integrated application is deployed. Make sure the Oracle WebLogic Server instance is part of the same WebLogic Server domain where OAAM is deployed.

4.1.1 Overview of the Integration Process

The high-level steps of the integration process is as follows:

1. Create a WebLogic Web application (`WAR`) or enterprise application (`ear`).
2. Add reference to the OAAM SDK Shared Library (`oracle.oaam.libs`) to the WebLogic deployment descriptor.
3. Implement the application that calls the OAAM APIs.
4. Add the application JAR files and other files.
5. Package the application, deploy it and test it.

4.1.2 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>
```

4.1.3 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>
```

4.1.4 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 `oaam_custom.properties` and place that file in the `WEB-INF\classes` directory of the native Web application.

For instructions on customizing, extending, or overriding Oracle Adaptive Access Manager properties, see Chapter 7, "Using the OAAM Extensions Shared Library to Customize OAAM."

4.2 OAAM Java In-Proc Integration

This section contains instructions to integrate OAAM using the In-Proc method.

1. Make sure you have set the reference to OAAM shared library `oracle.oaam.libs`.

   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>
   ```

   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>
   ```

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

This section contains instructions to integrate OAAM using the SOAP method. In the SOAP service wrapper API integration, the application communicates with Oracle Adaptive Access Manager using Web services.

Out-of-the-box, OAAM publishes Web services at the URL: `/oaam_server/services`. Starting with OAAM 11*g* Release 2 (11.1.2.0.0), the default mechanism to secure OAAM Web Services is by using Oracle Web Services Manager (OWSM) policies. Configuration of OWSM policies for authentication (HTTP Basic authentication with username and password request) and authorization (user's membership in configured group of users) is covered in this section. Authentication checks whether the passed user credentials are correct and authorization checks whether user is allowed to access the requested resource based on the user's membership in a group, for example, the user/group in the WebLogic embedded user store. Oracle Web Services Manager (OWSM) policies manage SOAP authentication and authorization through Oracle Enterprise Manager Fusion Middleware Control.

4.3.1 Enabling Web Services Authentication

OAAM Web Services can be protected by Oracle Web Services Manager (OWSM) using the policy `oracle/wss_http_token_service_policy`. The `wss_http_token_service_policy` policy enforces authentication and uses the credentials in the HTTP header to authenticate users. SOAP requests would be authenticated (HTTP Basic authentication) against the configured realm (users in WebLogic embedded user store).

To set up the Oracle Web Services Manager (OWSM) Policy to set HTTP Basic Authentication on `/oaam_server/services` follow these steps:

1. Log in to Oracle Enterprise Manager Fusion Middleware Control using the URL `http://weblogic-admin-hostname:port/em`.
2. Under **weblogic_domain**, select the domain and select **oaam_server_server1** and right-click and select the **Web Services** option.
3. Click **Attach Policies**.
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/no_authentication_service_policy** and **oracle/no_authorization_service_policy**.
   b. Click the **Next** button.

      If you disable the SOAP Web Service authentication on the server (which is by default enabled), the client can use the Web service without having been authenticated.
7. Click the **Attach** button in the next page.
8. Restart OAAM Server if required.

4.3.2 Creating User and Group

By performing the authentication configuration in this section, OAAM Web Services can be accessed by any valid username/password present in a configured realm, for

example, all the user credentials which can pass authentication, can access OAAM Web Services.

SOAP authentication is implemented using a user name and password. This user name and password must be associated with a user that is accessible to the application server. In order for that user to have permissions to perform operations on the web services, the user should be added to a group that can access the OAAM web services.

To secure web services, you must create a user with valid credentials who belongs to a group that can access URL: `/oaam_server/services`.

In a WebLogic deployment, this SOAP user can be stored and managed within the WebLogic security realm.

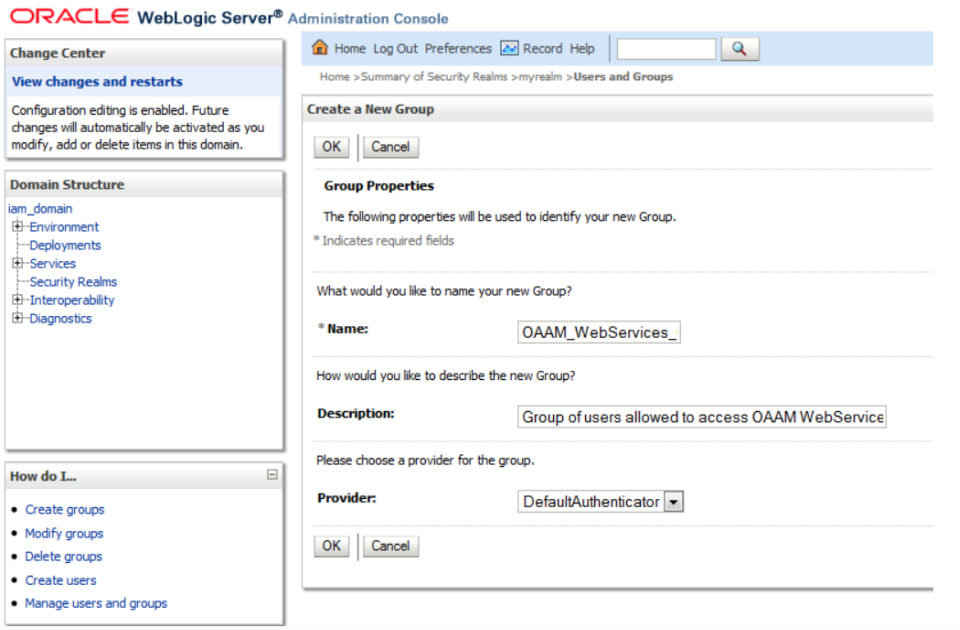
OAAM clients are configured to use this user name and password when invoking web services through the following `oaam_custom.properties` properties:

```
vcrypt.soap.auth.keystorePassword - Base64 encoded Password used to open the
   system_soap.keystore
vcrypt.soap.auth.aliasPassword - Base64 encoded Password used to retrieve the key
   stored in the keystore
vcrypt.soap.auth.username - Username of the SOAP user
vcrypt.soap.auth.keystoreFile -
   Filename of the keystore (should be system_soap.keystore)
```

To create the user and group, proceed as follows:

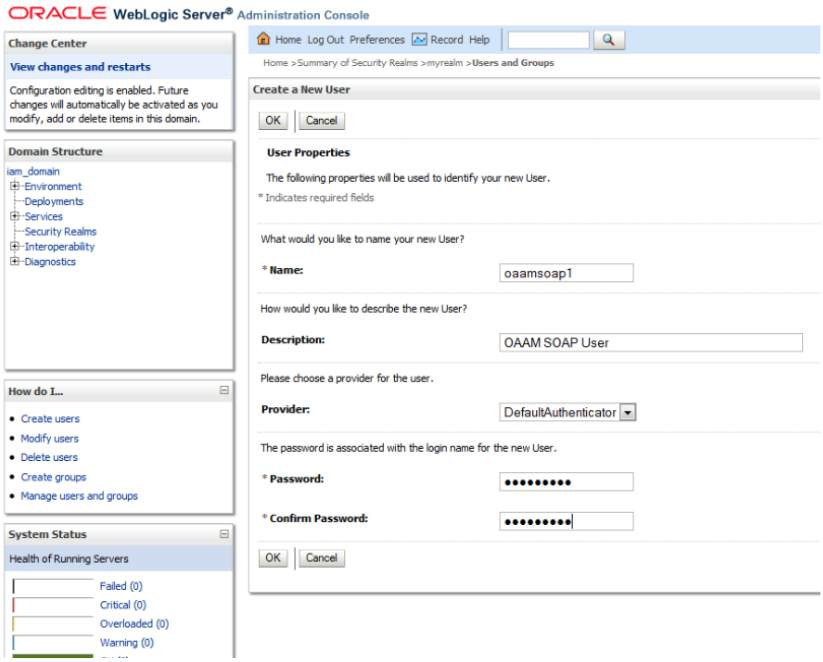
1. Using the WebLogic console, create a group in configured realm. For example, `OAAM_WebServices_Group`.

Figure 4–1 Create User Group



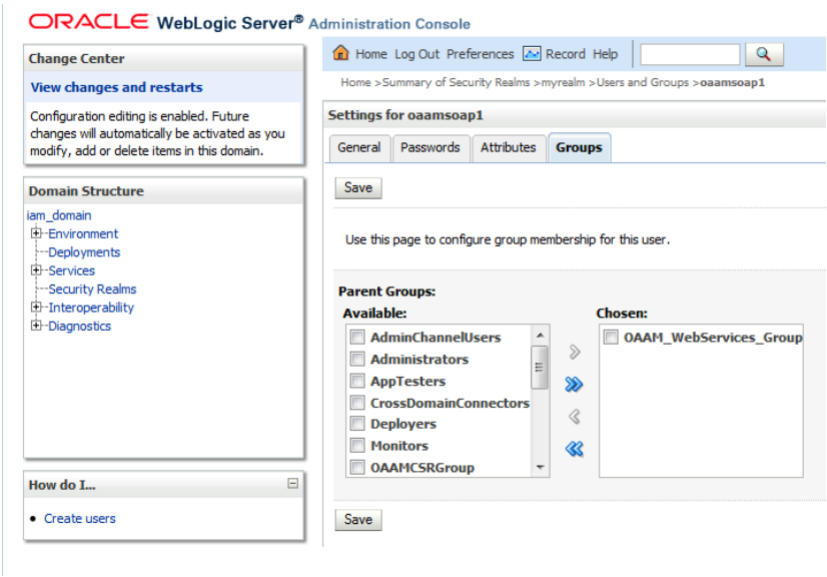
2. Create a user and associate the user with the group created in Step 1.

Figure 4–2 Creating a User and Associating the User with the Group



3. Update the user for group membership.

Figure 4–3 Configuring Group Membership for the User



4.3.3 Configuring Web Services Authorization

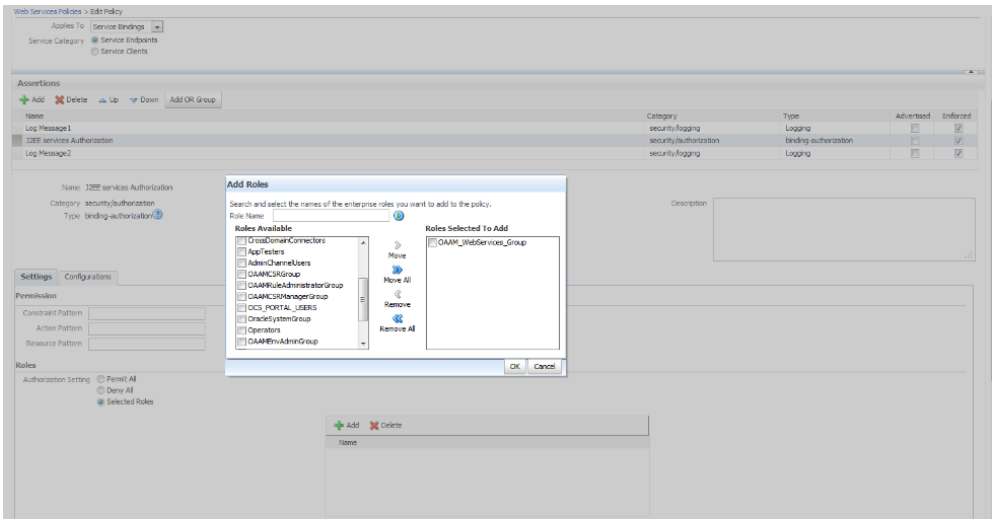
Using the Oracle Web Services Manager (OWSM) policy `oracle/binding_authorization_permitall_policy`, authorization can be configured for OAAM Web Services. The `binding_authorization_permitall_policy` policy provides simple permission-based authorization for the request based on the authenticated user at the SOAP binding level. This policy ensures that the user has permission to perform an operation. This policy should follow an authentication policy where the user is established and can be attached to Web Service Endpoints.

1. Log in to Oracle Enterprise Manager Fusion Middleware Control using the URL

   `http://weblogic-admin-hostname:port/em`

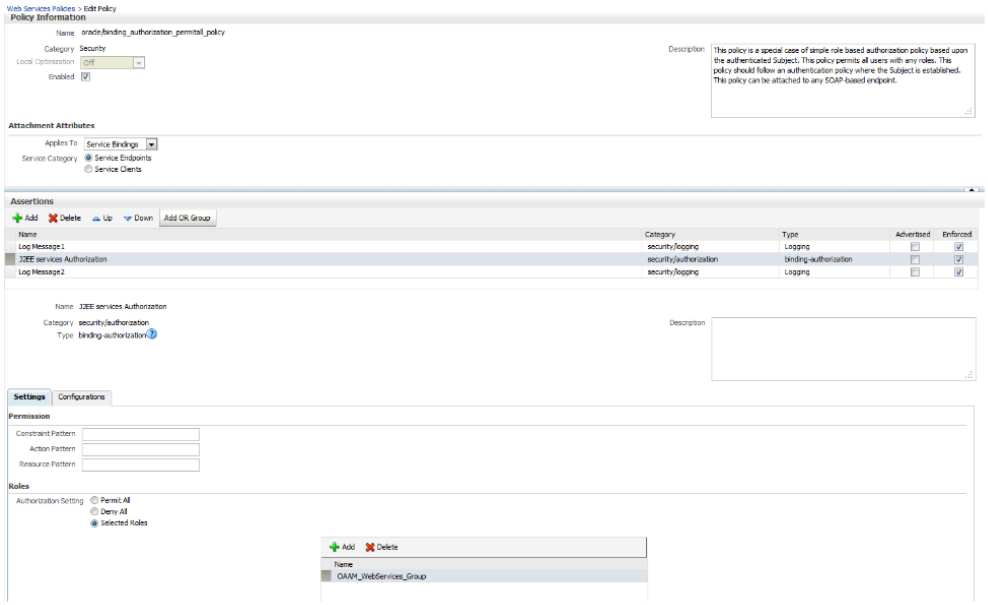
2. Expand the WebLogic Domain.
3. Right-click the domain hosting OAAM Server, **Web Services**, and **Policies**.
4. Select **oracle/binding_authorization_permitall_policy**.
5. Click **Edit**, and then the Settings tab.
6. Select **Selected Roles** from Authorization Setting.
7. Click Add (plus sign) and move the group to **Roles Selected To Add** list, and then click **OK**. The group was created in Section 4.3.2, "Creating User and Group."

Figure 4–4 User Role Added



8. Click **Save** to save the policy.

Figure 4–5 Role Added and Policy Saved.



9. To make sure that above policy configuration is working as expected, set property `active.protocol` to `remote`. The value for the property can be checked by navigating to domain hosting OAAM Server, right clicking Web Services, Platform Policy Configuration, and Policy Accessor Properties.

10. Attach the authorization policy to the Web Service Endpoints.

    Note: To get list of Web Service Endpoints exposed by OAAM Server on Enterprise Manager, go to Fusion Middleware Control, **Identity and Access**. Expand **OAAM**, then **oaam_server**, and right-click **Web Services**.

    a. Log in to Oracle Enterprise Manager Fusion Middleware Control using the URL

       `http://`*`weblogic-admin-hostname`*`:port/em`

    b. Under **weblogic_domain**, select the domain and select **oaam_server_server1** and right-click and select the **Web Services** option.

    c. Click **Attach Policies**.

    d. Select all the rows corresponding to OAAM Web Services and click the **Next** button

    e. Select the row **oracle/binding_authorization_permitall_policy**.

    f. Click the **Next** button.

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

    h. Restart OAAM Server if required.

4.3.4 Setting Up Client Side Keystore to Secure the SOAP User Password

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

To set up security for Native Client 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 OAAM Web Services Group role/group).
2. Copy sample.config_3des_input.properties to soap_3des_input.properties.

   ```
   cp sample.config_3des_input.properties 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. For example,
   #Welcome1
   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=

   keystorefilename=system_soap.keystore
   keystorealias=vcrypt.soap.call.passwd
   ```

4. Set ORACLE_MW_HOME and JAVA_HOME and source setCliEnv.sh.
5. 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==
   ```

6. Write down the Keystore password and Alias Password printed on the screen. You will need to add these to oaam_custom.properties.
7. Add the following properties with the encoded passwords (from step 5) and the authentication user name to oaam_custom.properties.

   OAAM clients are configured to use this user name and password when invoking web services through the following oaam_custom.properties properties:

   ```
   vcrypt.soap.auth.keystorePassword - Base64 encoded keystore password used to
   open the system_soap.keystore
   vcrypt.soap.auth.aliasPassword - Base64 encoded password to the alias used to
   retrieve the key stored in the keystore
   vcrypt.soap.auth.username - Username of the SOAP user configured for accessing
   the SOAP services
   vcrypt.soap.auth.keystoreFile - Filename of the keystore (should be system_
   ```

soap.keystore)

8. Save the system_soap.keystore file in your source code control system. Ensure you take adequate security precaution while handling this file. The file contains critical password information. Ensure 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 that is encrypted.
9. Copy your system_soap.keystore to *application*/WEB-INF/classes (classpath of the native client deployment).
10. Delete both the soap_key.file and soap_3des_input.properties files.

4.3.5 Setting SOAP Related Properties in oaam_custom.properties

Set the following properties in oaam_custom.properties of the native application:

```
vcrypt.common.util.vcryptsoap.impl.classname=
com.bharosa.vcrypt.common.impl.VCryptSOAPGenericImpl
vcrypt.tracker.impl.classname=
com.bharosa.vcrypt.tracker.impl.VCryptTrackerSOAPImpl
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.BharosaConfigLoadPropsImpl
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
```

4.3.6 Setting Up the Base Environment in OAAM Native SOAP Integration

The required JAR files for setting up the base environment in OAAM native SOAP integration are listed in this section. The following JAR files must be set in the JAVA classpath:

- jps-api.jar
- jps-common.jar
- jps-internal.jar

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 Oracle Adaptive Access Manager."

Note: isElementInList(), getListElements() and updateList() APIs do not support update/actions on the "alert group" lists.

4.5.1 addQuestion

addQuestion adds a new question for the specified user.

```
public boolean addQuestion(java.lang.String loginId, java.lang.String
questionText, java.lang.String answerText)
```

Table 4–1 addQuestion

Parameter	Description
loginId	The ID used by the user to login in
questionText	New question to be added. Overrides if the same question is already set for this user. Returns whether the operation was success or failure
answerText	Answer for the question

4.5.2 authenticatePassword

`authenticatePassword` authenticates the password.

```
public VCryptAuthResult authenticatePassword(java.lang.String loginId,
java.lang.String password, int authSessionType, int clientType, java.lang.String
clientVersion, java.lang.String ipAddress, int fingerPrintType, java.lang.String
fingerPrint)
```

Returns `VCryptAuthResult` object

Table 4–2 authenticatePassword

Parameter	Description
loginId	The ID used by the user to login in
password	New password to set
clientType	An enumeration value indicating the client type used for authentication
clientVersion	The version of the client; optional
authSessionType	Reason for authentication
ipAddress	IP address of the user device
fingerPrintType	Type of fingerprinting
fingerPrint	Fingerprint

4.5.3 authenticateQuestion

`authenticateQuestion` authenticates question or answer.

```
public VCryptAuthResult authenticateQuestion(java.lang.String loginId,
java.lang.Long authSessionId, java.lang.String answer, java.lang.String ipAddress,
int fingerPrintType, java.lang.String fingerPrint)
```

Returns `VCryptAuthResult` describing result of authentication attempt.

Table 4–3 authenticateQuestion

Parameter	Description
loginId	The ID used by the user to authenticate answer
authSessionId	ID of the authentication session
answer	The answer given by the user
ipAddress	IP address of the user device
fingerPrintType	Type of fingerprinting
fingerPrint	Fingerprint

4.5.4 cancelAllTemporaryAllows

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

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

Table 4–4 cancelAllTemporaryAllows Parameters

Parameter	Description
customerId	The customer ID

4.5.5 clearSafeDeviceList

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

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

Table 4–5 clearSafeDeviceList Parameters

Parameter	Description
requestId	The ID for the login session. The same ID is necessary for all the calls to Bharosa API for the login session.

4.5.6 createOAAMSession

A Session ID is required for the creation and update of transactions. If a Session ID is not available, you must call the `createOAAMSession` API to create the OAAM session. After obtaining the Session ID from the session, you can call the `CreateTransaction` API to create a transaction.

When you create a session, you specify values in the `createOAAMSession` request and then call the API.

```
createOAAMSession(String requestId, Date requestTime, OAAMUserData user,
OAAMIPData ip,
```

```
List<OAAMDeviceFingerprintData> cookieList, OAAMSessionData sessionData)
```

Table 4–6 createOAAMSession Parameters

Parameter	Description
requestId	requestId identifies the user session; If requestId is not supplied, OAAM server will create one; however, if requestId is supplied and then if the OAAM server cannot find one, then an error is returned.
requestTime	requestTime is the time of the request; it can be null; if null, the server uses the current time
user	User is the OAAM user
ip	The user session's IP address
cookieList	List of cookies stored in the session
sessionData	Data from the session

4.5.7 createOrUpdateEntities

You can use the `createOrUpdateEntities` API to perform the following tasks:

- Create and update entities
- Replace and merge attribute values during an entity update

```
public VCryptObjectResponse<VCryptObjectResponse<EntityHeader>[]>
createOrUpdateEntities(EntityData[] entityRequestData,boolean isReplaceEntity, int
commitBatchSize, String requestId);
```

Table 4–7 Create or Update Entities API

Parameter	Description
entityRequestData	Array of EntityData objects. An EntityData object contains the information required to create one entity. For details on EntityData.java, see the *Oracle Fusion Middleware Java API Reference for Oracle Adaptive Access Manager*.
isReplaceEntity	Flag to determine replacement or merging of attributes on update of entity. Default value: `FALSE` which denotes merge.
commitBatchSize	Determines the number of entities which must be committed together. Default and minimum value is `1`
requestId	Value to identify the session. The value is sent by the client. If the client does not set this value then OAAM generates a dummy number
VCryptObjectResponse:	`SUCCESS` on successful execution of API (there is no database or connection error) and at least one entity is created `response.getObject()` returns Array object containing `VCryptObjectResponse` for individual entities. Each response object contains an `EntityHeader` object on `SUCCESS`. Query for `response.isSuccess()`. (`true` for `SUCCESS` and `false` for `ERROR`). `ERROR` if no entity is created. `response.getObject()` returns object containing `VCryptObjectResponse`. Each response object contains error message on `ERROR`.

4.5.8 createTransaction

A Session ID is required for the creation of transactions. If a Session ID is not available, you must call the `createOAAMSession` API to create the OAAM session. After

obtaining the Session ID from the session, you can call the `createTransaction` API to create the transaction.

createTransaction creates a new transaction.

```
public VCryptResponse createTransaction(
TransactionCreateRequestData trxUpdData =
               new TransactionCreateRequestData(sessionId,
                                                requestTime,
                                                transactionDefKey,
                                                externalTransactionId,
                                                trxStatus, trxDataMap,
                                                analyzePatterns);
               response =

VCryptTrackerUtil.getVCryptTrackerInstance().createTransaction(trxUpdData);



    TransactionResponse transResponse = response.getTransactionResponse();
        Long transId = null;
        if (transResponse != null){
            transId = transResponse.getTransactionId();
        }
```

Table 4–8 createTransaction Parameter and Returned Value

Parameter	Description
TransactionCreateRequestData	The object to create a transaction; it throws the exception `BharosaException` if it fails validation. The structure of this object is as follows: ■ `sessionID` identifies the user session; required. A Session ID is required for the creation of transactions. If a Session ID is not available, you must call the createOAAMSession API to create the OAAM session. ■ `requestTime` is the time of the request; can be null; if null, the server uses the current time ■ `transactionDefKey` 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. ■ `trxStatus` is the transaction status; can be null. The corresponding enum name is `tracker.transaction.status.enum`. ■ `trxDataMap` 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` ■ 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`."
VCryptResponse	The response object; make sure to check `isSuccess()` before obtaining the transaction ID with the method `getTransactionResponse()`

4.5.9 createUser

`createUser` creates an user in the authentication database.

```
public VCryptAuthUser createUser (VCryptAuthUser user)
```

Table 4–9 createUser

Parameter	Description
User	Authenticated user. Returns the newly created authenticated user.

4.5.10 deleteQuestion

`deleteQuestion` deletes the question for the specified user.

```
public boolean deleteQuestion(java.lang.String loginId, java.lang.String question)
```

Table 4–10 deleteQuestion

Parameter	Description
loginId	The user login ID
question	The question to be deleted. Returns whether the operation was success or failure.

4.5.11 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 4–11 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` property must be set to `true`.

4.5.12 getCaption

`getCaption` gets a caption for the user.

```
public java.lang.String getCaption(java.lang.String loginId)
```

Table 4–12 getCaption

Parameter	Description
loginId	The login id of the user. Returns caption string

4.5.13 getFinalAuthStatus

`getFinalAuthStatus` returns the final authentication status of a user. The status can be no more than 30 days old.

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

Table 4–13 getFinalAuthStatus Parameters

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

4.5.14 getImage

`getImage` gets the `imagePath` for the user.

```
public java.lang.String getImage(java.lang.String loginId)
```

Table 4–14 getImage

Parameter	Description
loginId	The login ID of the user. Returns path to the image

4.5.15 getOTPCode

The generateOTP() API has been deprecated in the OAAM JAVA and SOAP APIs. Please use the getOTPCode() API instead when writing your production code. For details on how to use the getOTPCode() API, see the *Oracle Fusion Middleware Java API Reference for Oracle Adaptive Access Manager*.

4.5.16 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 4–15 getRulesData Parameters

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

4.5.17 getSecretQuestion

`getSecretQuestion` gets a secret question for the user.

```
public VCryptQuestion getSecretQuestion(java.lang.String loginId)
```

Table 4–16 getSecretQuestion

Parameter	Description
loginId	The login ID of the user to authenticate. Returns object containing the question to ask

4.5.18 getSignOnQuestions

`getSignOnQuestions` gets all the secret questions available for the user.

```
public VCryptQuestion getSignOnQuestions(java.lang.String loginId)
```

Table 4–17 getSignOnQuestions

Parameter	Description
loginId	The login ID of the user to authenticate. Returns the 2-D array object containing the questions to ask. First dimension denotes the number of (configurable) question sets to display to user and the second dimension denotes the number of questions in each question set.

4.5.19 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 4–18 getUserByLoginId

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

4.5.20 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 4–19 handleTrackerRequest Parameters

Parameter	Description
requestId	The login session ID; this is the ID that is necessary 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 the 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

4.5.21 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 4–20 handleTransactionLog Parameters

Parameter	Description
requestId	The login session ID; this is the ID that is necessary 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

4.5.22 IsDeviceMarkedSafe

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

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

Table 4–21 IsDeviceMarkedSafe Parameters

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

4.5.23 markDeviceSafe

`markDeviceSafe` marks the user device as safe.

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

Table 4–22 markDeviceSafe Parameters

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

4.5.24 processPatternAnalysis

`processPatternAnalysis` triggers the data pattern processing.

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

Table 4–23 processPatternAnalysis Parameters

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

Table 4–23 (Cont.) processPatternAnalysis Parameters

Parameter	Description
transactionId	The identifier of the transaction. For authentication type of data can pass 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.

4.5.25 processRules

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.

`processRules` processes policy sets for the passed checkpoints.

```
VCryptRulesResult ruleResult =
    VCryptTrackerUtil.getVCryptRulesEngineInstance().processRules(
                                    sessionId,
                                    transId,
                                    externalTransactionId,
                                    requestTime,
                                    runtimeList,
                                    contextDataMap);
```

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

Table 4–24 processRules Parameters

Parameter	Description
sessionId	The login session ID; this is the ID that is necessary in all API calls for the login session
transId	The transaction session ID; this is the ID that is necessary in all API calls for the transaction session
externalTransactionId	`externalTransactionId` is used to correlate the application transaction with the corresponding OAAM Transaction. It can also be used to update the transaction.
requestTime	The time at which the request was made
runtimeTypes	The list of checkpoints to be evaluated; each checkpoint in this list is evaluated. The runtimeTypes is a singleton list of Integer type. For example, to run a pre-transaction checkpoint, create the following list: `List PRE_TRANSACTION_RUNTIME_LIST = Collections.singletonList(new Integer(1));`
contextDataMap	A list of key-value pairs identifying the context data. The `contextDataMap` in this API is to provide any additional parameters that are needed for rules processing. For example, to verify if the IP has changed during the session, then you can provide the IP in the `contextDataMap` of the `Process Rules` API and OAAM can compare the new IP against the original IP that was used during create/update transaction.

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, ensure 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) ;
}
```

4.5.26 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 4–25 resetUser Parameters

Parameter	Description
customerId	The customer ID

4.5.27 searchEntityByKey

You can use the `searchEntityByKey` API to find entities based on its key attributes.

```
public VCryptObjectResponse<EntityHeader>
   searchEntityByKey(EntityData entityData);
```

Table 4–26 searchEntityByKey

Parameter	Description
entityData	EntityData object with `entityName` and `entityDataMap` containing key(s) and value(s) of primary key attributes of the entity to be searched based on the ID scheme
VCryptObjectResponse	Contains EntityHeader object which is the entity object on `SUCCESS` or `error` message on `ERROR`

4.5.28 setCaption

`setCaption` sets a new caption for the specified user.

```
public boolean setCaption(java.lang.String loginId, java.lang.String caption)
```

Table 4–27 setCaption

Parameter	Description
loginId	The login ID of the user
caption	New caption to set. Returns whether the operation was success or failure

4.5.29 setImage

`setImage` sets a new image for the user.

```
public boolean setImage(java.lang.String loginId, java.lang.String imagePath)
```

Returns whether the operation was success or failure

Table 4–28 setImage

Parameter	Description
loginId	The login ID of the user
imagePath	Path to the image file.

4.5.30 setPassword

`setPassword` sets a new password for the specified user.

```
public boolean setPassword(java.lang.String loginId, java.lang.String password,
int passwordStatus)
```

Returns whether the operation was success or failure

Table 4–29 setPassword

Parameter	Description
loginId	The login ID of the user
password	New password to set
passwordStatus	Status of the password

4.5.31 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 4–30 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

4.5.32 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; ensure 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 you can only use the values of this enumeration 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 4–31 updateAuthStatus Parameters

Parameter	Description
requestId	The login session ID; this is the ID that is necessary 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."

4.5.33 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 4–32 updateLog Parameters

Parameter	Description
requestId	The login session ID; this is the ID that is necessary 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

Table 4–32 (Cont.) updateLog Parameters

Parameter	Description
fingerPrintType	Refer to the OAAM enum `vcrypt.fingerprint.type.enum` for a list of valid values. Currently the enum has the 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.
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 the 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

4.5.34 updateTransaction

Both Session ID and Transaction ID are required to update transactions. If a Session ID is not available, you must call the `createOAAMSession` API to create the OAAM session. The Session ID is required by the `createTransaction` API. You must call the `createTransaction` API to create a Transaction ID before you can call the `updateTransaction` API to update the transaction.

`updateTransaction` updates a previously created transaction.

```
    TransactionUpdateRequestData trxUpdData =
new TransactionUpdateRequestData(sessionId,
 requestTime,
 transactionId,
 new Integer(trxStatus),
 trxDataMap,
 Boolean.TRUE);
    response =
    VCryptTrackerUtil.getVCryptTrackerInstance().updateTransaction(trxUpdData);
```

Table 4–33 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 causes the exception `BharosaException` if it fails validation. The structure of this object is as follows: ■ `sessionId` identifies the user session; required ■ `requestTime` is the time of the request; can be null; if null, the server uses the current time ■ `transactionId`, the ID returned by a previous call to `createTransaction` ■ `trxStatus`, the status of the transaction. The possible values are as follows: ■ success=1 ■ block=2 ■ reject=3 ■ wrong_answer=4 ■ pending=99 ■ `trxDataMap` 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`.
VCryptTrackerInstance	The response object; make sure to check `isSuccess()` before obtaining the transaction ID with the method `getVCryptTrackerInstance()`

4.5.35 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 4–34 updateTransactionStatus Parameters

Parameter	Description
requestId	The login session ID; this is the ID that is necessary 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."

5

Creating, Updating, and Searching for Entities Using the Entity API

Entities are data structures that can be associated as an instance of a transaction. Any process a user performs after successfully logging in can be termed as a transaction. Oracle Adaptive Access Manager can evaluate the risk associated with a transaction in real-time to prevent fraud and misuse.

This chapter explains how applications can use the Entity API to create, update, and search for entities. It contains these sections:

5.1 About the Entity APIs

OAAM provides two Entity APIs that allow applications to manage entities and entity relationships needed to facilitate fraud detection.

5.1.1 Entity Tasks

The Entity APIs allow applications to perform create and update, replace, and search operations on entities in the database. The Entity APIs can:

- Create and update an entity in the OAAM database schema
- Replace or merge entity attributes
- Search for entities

Entity tasks performed on the client's transaction data in the database require the following information:

- The Entity key is the key provided by the Administrator when creating the entity definition in OAAM Admin
- Entity data is the data entered by the user of the client application
- The linked entity relationship name is the name specified by the Administrator when creating an entity definition in OAAM Admin. A linked entity relationship name is required for complex entity instances only.

5.1.2 Processing Status

VCryptObjectResponse contains information about the status of the processing.

Entity Create and Update API Return type: VCryptObjectResponse:

Result	Description
SUCCESS	SUCCESS on successful execution of API (there is no database or connection error) and at least one entity is created. response.getObject() returns Array object containing VCryptObjectResponse for individual entities. Each response object contains an EntityHeader object on SUCCESS. Query for response.isSuccess(). (true for SUCCESS and false for ERROR).
ERROR	ERROR if no entity is created. response.getObject() returns object containing VCryptObjectResponse. Each response object contains error message on ERROR.

Entity Search API Return type: VCryptObjectResponse

Return Object	Description
VCryptObjectResponse	Contains EntityHeader object which is the entity object on SUCCESS or error message on ERROR

5.1.3 Create or Update Entities

You can use the createOrUpdateEntities API to perform the following tasks:

- Create and update entities
- Replace and merge attribute values during an entity update

When you create an entity, a unique key is generated using the primary key values from the entity data map. OAAM uses that key to check if the entity already exists in the database. If the entity does not exists, an entity is created, otherwise the entity is updated. The entity is updated based on the data specified in the entity data map.

To erase values for some attributes of an existing entity using the entityId, populate entityDataMap as follows, and set the isReplaceEntity (In createOrUpdateEntity) parameter to true.

API Signature:

```
public VCryptObjectResponse<VCryptObjectResponse<EntityHeader>[]>
createOrUpdateEntities(EntityData[] entityRequestData,boolean
isReplaceEntity, int commitBatchSize, String requestId);
```

Parameters:

Parameter	Description
entityRequestData	Array of EntityData objects. An EntityData object contains the information required to create one entity. For details on EntityData.java, see the *Oracle Fusion Middleware Java API Reference for Oracle Adaptive Access Manager*, Release 11g.
isReplaceEntity	Flag to determine replacement or merging of attributes on update of entity. Default value: FALSE which denotes merge.
commitBatchSize	Determines the number of entities which must be committed together. Default and minimum value is 1

Parameter	Description
requestId	Value to identify the session. The value is sent by the client. If the client does not set this value then generate a dummy number

5.1.4 Replace or Merge Attributes

Replacing or Merging Attributes: During an entity update, you can choose to merge or replace attributes. The difference between the two modes is visible only in the case where the user passes empty values for attributes in entity data. In the `merge` case, the value of such attributes will not change and the old value persists. In the `replace` case, the empty value will replace the old value and such attributes will be replaced with empty values. Default option is `merge`.

5.1.5 Search Entity By Key

You can use the `searchEntityByKey` API to find entities based on its key attributes.

API Signature:

```
public VCryptObjectResponse<EntityHeader>
searchEntityByKey(EntityData entityData);
```

Parameters:

Parameter	Description
entityData	`EntityData` object with `entityName` and `entityDataMap` containing key(s) and value(s) of primary key attributes of the entity to be searched based on the ID scheme

5.2 Creating Entities and Mapping Attributes

This section contains information about mapping attributes for entity resolution in fraud detection.

5.2.1 Entity Data Map

OAAM uses the entity data map to create an entity. The entity map includes the key value pairs of attribute names and their values specified by the user. For instance, for creating an entity of type `Customer` the data map can be:

Entity data map:

Key: first name	Value: Mark
Key: last name	Value: Smith
Key: email	Value: x@y.com
Key: shipping.addr_line1	Value: #1, Lex Residence
Key: shipping.addr_line2	Value: Redmond Street
Key: shipping.zip	Value: 418001
Key: shipping.phone_number	Value: 6035550100

5.2.2 Complex Entity

A complex entity has other entities linked to it by a relationship name. For instance a `Customer` can be defined by the attributes listed in the subsequent example. Other entities link to it by a relationship name.

To receive data for creating a complex entity a dot convention is used for the keys in the entity data map. The attribute names of the linked entities in the data map must be preceded by the relationship name and a dot(.).

Example:

An attribute of an entity can be an entity itself. Such an attribute is called a linked entity. For instance, `Shipping address` is a linked entity of the entity, `Customer`.

```
Customer:     first name (Simple attribute)
                  last name (Simple attribute)
                  phone (Simple attribute)
                  email (Simple attribute)
                  shipping address:           addr_line1
                                  :           addr_line2
                                  :           city
                                  :           state
                                  :           country
                                  :           mobile
                                  :           zip
                  billing address :           addr_line1
                                  :           addr_line2
                                  :           city
                                  :           state
                                  :           country
                                  :           mobile
                                  :           zip
```

5.2.3 Creating a Simple Entity

A simple entity is one which is not linked to any other entity. When you create an entity instance, the entity related data is stored in the database.

The `EntityData` object takes following parameters:

Entity name: The entity name determines the type of entity to be created. For example, `Customer`. The definition for this entity type already exists in the database for entity creation.

Entity data map: The entity data map includes the key value pairs of attribute names and their values specified by the user. To create a simple entity that does not have any related entities, populate `entityDataMap` as follows:

Key	Value
Key: *attributeName1*	Value: *attributeValue1*
Key: *attributeName2*	Value: *attributeValue2*
Key: *attributeName3*	Value: *attributeValue3*

For example: create a `Customer` entity that does not have any related entity

Context Data	Values
Key: first name	Value: Mark
Key: last name	Value: Smith
Key: email	Value: x@y.com
Key: mobile	Value: 6035550100

To create a simple entity instance:

1. Create a map that contains the entity data.

   The map key is the entity attribute name (as specified in OAAM Admin), and the map value is the user input value.

   In the subsequent example, you can see data for the `Customer` entity data, and user input is `Mark Smith` as name. The `Customer` entity has `first name` and `last name` attributes.

   ```
   Map<String,String> entityDataMap = new HashMap<String,String>();
   entityDataMap.put("first name","Mark");
   entityDataMap.put("last name","Smith");
   ```

2. Create an `EntityData` object that encapsulates the entity data map and the entity type. For example, `Customer`.

   ```
   EntityData entityData= new EntityData("Customer",entityDataMap);
   ```

3. Since the API only accepts array of EntityData's, create an array and insert the `entitydata` created into it.

   ```
   EntityData[] entityRequestData= new EntityData[1];
   entityRequestData[0]= entityData;
   ```

4. Finally, call the API `tracker`. This is a `VCryptTracker` instance.

   ```
   response =
   tracker.createOrUpdateEntities(entityRequestData, true,
   commitBatchSize,requestId);
   ```

Errors occur if you try to create an entity instance in the following ways:

- With an entity that does not exist
- With null as the entity name
- Without providing the required information
- With null values as the required information
- Using mismatching entity data types
- Using entirely different entity data as compared to entity definition

5.2.4 Updating Attributes of an Existing Entity

Use the `createOrUpdateEntities` API to update attributes of an existing entity.

entityId (id)

`id` is the value of the Entity ID as stored in the database. You must pass the values of all the key attributes of the entity. This would uniquely identify the entity instance in the database.

Obtain the `entityId` (`$id$`) from the response of `createOrUpdateEntity` API. For example:

```
VCryptObjectResponse<EntityHeader>[] responseArray = response.getObject();
VCryptObjectResponse<EntityHeader> entityResponse = responseArray[0];

EntityHeader entityHeader = entityResponse.getObject();
Long entityId= entityHeader.getEntityId();
```

To update some attributes of an existing entity using the `entityId`, populate `entityDataMap` as follows:

entityDataMap:

Key: id	**Value: *EntityId***
Key: *attributeName1*	Value: *attributeValue1*
Key: *attributeName2*	Value: *attributeValue2*

For example, update the email Id of a `Customer` entity with entityId as `101`:

Key: id	**Value: 101**
Key: email	Value: a@b.com

There are two ways to update an entity instance:

For example:

```
Customer (101):     firstname           ->          Mark
                    lastname            ->          Smith
                    mobile              ->          0987654321
                    email               ->          x@y.com
```

Suppose the prior example is an entity of type `Customer` with entity ID `101`. Suppose you want to change its email to `a@b.com`, there are two way to pass entity data.

The first way is shown as:

```
                    $id$                ->          101
                    email               ->          a@b.com
```

The second way is shown as:

```
                   firstname            ->          Mark
                   lastname             ->          Smith
                   email                ->          a@b.com
```

5.2.5 Erasing the Value of Attributes of an Existing Entity

To erase values for some attributes of an existing entity using the `entityId`, populate `entityDataMap` as follows, and set the `isReplaceEntity` (In `createOrUpdateEntity`) parameter to `true`.

entityDataMap:

Key: id	**Value: *EntityId***
Key: *attributeName1*	Value:
Key: *attributeName2*	Value:

For example, erase the email Id of a `Customer` entity with `entityId` as `101`:

Key: id	**Value: 101**
Key: email	

5.2.6 Creating an Entity that has Related Entities with Complete Data of Both Top-Level Entity and Related Entities

To create an entity that has related entities with complete data of both top-level entity and related entities, use this format:

entityDataMap:

Key: *attributeName1*	**Value: *attributeValue1***
Key: *attributeName2*	Value: *attributeValue2*
Key: *attributeName3*	Value: *attributeValue3*
Key: *relationshipName1.attributeName1*	Value: *linkedEnt1AttributeValue1*
Key: *relationshipName1.attributeName2*	Value: *linkedEnt1AttributeValue2*
Key: *relationshipName2.attributeName1*	Value: *linkedEnt2AttributeValue1*

Shipping is a relationship name which links `Customer` to another entity of type `address`. To receive data for creating a complex entity, a dot convention for the keys in the entity data map. The attribute names of the linked entities in the data map must be preceded by the relationship name and a dot(.).

Example: to create a `Customer` entity with linked address entities with `relationshipNames` as shipping and billing

Key: first name	**Value: Mark**
Key: last name	Value: Smith
Key: email	Value: x@y.com
Key: shipping.addr_line1	Value: #1, Lex Residence
Key: shipping.addr_line2	Value: Redmond Street
Key: billing.addr_line1	Value: #2, Lex Residence

5.2.7 Creating an Entity that has Related Entities (with Multiple Instances of a Single Entity) with Complete Data of Both Top-Level Entity and Related Entities

To create an entity that has related entities (with multiple instances of single relationships) with complete data of both top-level entity and related entities, use the `createOrUpdateEntities`() API.

entityDataMap:

Key: *attributeName1*	**Value: *attributeValue1***
Key: *attributeName2*	Value: *atributeValue2*
Key: *attributeName3*	Value: *attributeValue3*
Key: *relationshipName1*[*index1*].*attributeName1*	Value: *linkedEnt1AttributeValue1*
Key: *relationshipName1*[*index1*].*attributeName2*	Value: *linkedEnt1AttributeValue2*
Key: *relationshipName1*[*index2*].*attributeName1*	Value: *linkedEnt2AttributeValue1*
Key: *relationshipName1*[*index2*].*attributeName2*	Value: *linkedEnt2AttributeValue2*
Key: *relationshipName2.attributeName1*	Value: *linkedEnt3AttributeValue1*

Example: to create a `Customer` entity with linked address entities with `relationshipNames` as shipping and billing with two instance of shipping.

Key: first name	**Value: Mark**
Key: last name	Value: Smith
Key: email	Value: x@y.com
Key: shipping[0].addr_line1	Value: #1, Lex Residence
Key: shipping[0].addr_line2	Value: Redmond Street
Key: shipping[1].addr_line1	Value: #3, Lex Residence
Key: shipping[1].addr_line2	Value: Redwood Street
Key: billing.addr_line1	Value: #2, Lex Residence

5.2.8 Creating an Entity that has Related Entities with Complete Data of Top-level Entity and Entity Ids of One or More Related Entities

To create an entity that has related entities with complete data of top-level entity and entity Ids of one or more related entities

entityDataMap:

Key: *attributeName1*	**Value: *attributeValue1***
Key: *attributeName2*	Value: *attributeValue2*
Key: *attributeName3*	Value: *attributeValue3*
Key: *relationshipName1*[*index1*].*attributeName1*	Value: *linkedEnt1AttributeValue1*
Key: *relationshipName1*[*index1*].*attributeName2*	Value: *linkedEnt1AttributeValue2*
Key: *relationshipName1*[*index2*].id	Value: *linkedEnt2EntityId*
Key: *relationshipName2.attributeName1*	Value: *linkedEnt3AttributeValue1*

Example: to create a `Customer` entity with linked `address` entities with `relationshipNames` as `shipping` and `billing` with two instance of shipping. One of the `shipping address` entity already exists with `entityId` as `102`.

Key: first name	**Value: Mark**
Key: last name	Value: Smith

Key: first name	Value: Mark
Key: email	Value: x@y.com
Key: shipping[0].addr_line1	Value: #1, Lex Residence
Key: shipping[0].addr_line2	Value: Redmond Street
Key: shipping[1].id	Value: 102
Key: billing.addr_line1	Value: #2, Lex Residence

Errors occur if you try to create a linked entity instance in the following ways:

- Using a non-existent linked entity name
- Using an empty linked entity name
- Without all the required linked entity data
- With null values for the required linked entity data
- Using mismatching linked entity data data type
- Using entirely different linked entity data as compared to the entity definition
- Where the linked entity is of the same type as the parent

If you create a complex entity instance where the linked entity (multilevel) is of the same type as the parent, the entity instance is created with an error status. The error is from the mismatch in the entity definition since such a definition is not allowed.

5.2.9 Updating Related Entities of an Entity with Entity Ids of Related Entities

To update related entities of an entity with entity Ids of related entities

entityDataMap:

Key: ***attributeName1***	Value: ***attributeValue1***
Key: *attributeName2*	Value: *attributeValue2*
Key: *relationshipName1*.id	Value: *linkedEnt1EntityId*
Key: *relationshipName1.attributeName1*	Value: *linkedEnt1AttributeValue1*

Note: One can also pass the parent `entityId` instead of attributes for the parent entity.

Example: to update the city to `Chicago` in billing address for Customer `Mark Smith`. The billing address already exists with `entityId` as `103`

Key: first name	Value: Mark
Key: last name	Value: Smith
Key: billing.id	Value: 103
Key: billing.city	Value: Chicago

5.2.10 Unlinking Linked Entities.

You can Unlink one or more related entities from the parent entity.

```
Customer:          first name:: abc
                 : last name:: xyz
```

```
: mobile:: 6035550100
: email:: p@q.com
: shipping address:: (entity with id = 102)
: shipping address:: (entity with id = 105)
: billing address:: (entity with id = 103)
```

unLinkEntities: List of mapId's of relationships to be deleted. mapId's is of type `Long`. The mapIds can be fetched from list of `VTEntityOneMap` objects returned as `linkedEntities` in Entity object.

unlinkEntities:

Value (In form of a List)
102,105
103

If unlinkEntities is specified as `null` while updating entity instances, then no changes are made to the existing entity relationships. All the entity relationships previously associated with the parent entity involved in update operation persist. New relationships can be however added (using entity data map) during the update.

An error occurs if you try to unlink entities in the following ways:

- By passing in required attributes that do not correspond to existing the entity instance.
- By passing in entity Id values that are null/empty
- By passing in a parent entity Id, thereby removing the required attributes
- By passing in entity Id values that do not exist
- By passing in duplicate entity Id

Code Example

```
public void testDeleteRelationships() throws Exception{
    boolean isReplaceEntity = false;
    int commitBatchSize=1;
    String timeStamp = Long.toString(System.currentTimeMillis());
    EntityData[] entityRequestData= new EntityData[1];
    Map<String,String> entityDataMap = new HashMap<String,String>();
         entityDataMap.put("first name","Mark"+timeStamp);
         entityDataMap.put("last name","Smith"+timeStamp);
         entityDataMap.put("email","x@y.com");
         entityDataMap.put("mobile","6035550100");
    EntityData entityData= new EntityData("customer",entityDataMap);
    entityRequestData[0]= entityData;
    String requestId= null;
    VCryptObjectResponse<VCryptObjectResponse<EntityHeader>[]> response =
vCryptTracker.createOrUpdateEntities(entityRequestData, isReplaceEntity,
commitBatchSize,requestId);
    assertTrue(response.isSuccess());
    VCryptObjectResponse<EntityHeader>[] responseArray= response.getObject();
    VCryptObjectResponse<EntityHeader> entityResponse= responseArray[0];
    assertTrue(entityResponse.isSuccess());
    EntityHeader entity = entityResponse.getObject();
    Long customerEntityId= entity.getEntityId();

    // creating an address
```

```
    Map<String,String> entityDataMapAddress1 = new HashMap<String,String>();
         entityDataMapAddress1.put("addr_line1","testHouse1b"+timeStamp);
         entityDataMapAddress1.put("addr_line2","testStreet1b");
         entityDataMapAddress1.put("addr_line3","testlane1b");
         entityDataMapAddress1.put("city","city1");
         entityDataMapAddress1.put("state","state1");
         entityDataMapAddress1.put("country","country1");
         entityDataMapAddress1.put("zip","333031");
         entityDataMapAddress1.put("phone","6035550100");
    EntityData entityDataAddress1= new
    EntityData("address",entityDataMapAddress1);
    entityRequestData[0]= entityDataAddress1;
    VCryptObjectResponse<VCryptObjectResponse<EntityHeader>[]> responseAddress1 =
vCryptTracker.createOrUpdateEntities(entityRequestData, isReplaceEntity,
commitBatchSize,requestId);
    assertTrue(responseAddress1.isSuccess());
    VCryptObjectResponse<EntityHeader>[] responseArrayAddress1=
responseAddress1.getObject();
    VCryptObjectResponse<EntityHeader> entityResponseAddress1=
responseArrayAddress1[0];
    assertTrue(entityResponseAddress1.isSuccess());
    EntityHeader entityAddress1 = entityResponseAddress1.getObject();
    Long address1EntityId= entityAddress1.getEntityId();

    // creating another address
    Map<String,String> entityDataMapAddress2 = new HashMap<String,String>();
    entityDataMapAddress2.put("addr_line1","testHouse2"+timeStamp);
    entityDataMapAddress2.put("addr_line2","testStreet2");
    entityDataMapAddress2.put("addr_line3","testlane2");
    entityDataMapAddress2.put("city","city2");
    entityDataMapAddress2.put("state","state2");
    entityDataMapAddress2.put("country","country2");
    entityDataMapAddress2.put("zip","333031");
    entityDataMapAddress2.put("phone","6035550100");
    EntityData entityDataAddress2= new
    EntityData("address",entityDataMapAddress2);
    entityRequestData[0]= entityDataAddress2;
    VCryptObjectResponse<VCryptObjectResponse<EntityHeader>[]> responseAddress2 =
vCryptTracker.createOrUpdateEntities(entityRequestData, isReplaceEntity,
commitBatchSize,requestId);
    assertTrue(responseAddress2.isSuccess());
    VCryptObjectResponse<EntityHeader>[] responseArrayAddress2=
responseAddress2.getObject();
    VCryptObjectResponse<EntityHeader> entityResponseAddress2=
responseArrayAddress2[0];
    assertTrue(entityResponseAddress2.isSuccess());
    EntityHeader entityAddress2 = entityResponseAddress2.getObject();
    Long address2EntityId= entityAddress2.getEntityId();

    // creating relationships between the customer and addresses
    Map<String,String> entityDataMapRelation = new HashMap<String,String>();
    entityDataMapRelation.put("$id$",customerEntityId.toString());
    entityDataMapRelation.put("shipping.$id$",address1EntityId.toString());
    entityDataMapRelation.put("billing[0].$id$",address1EntityId.toString());
    entityDataMapRelation.put("billing[1].$id$",address2EntityId.toString());
    EntityData entityDataRelation= new
EntityData("customer",entityDataMapRelation);
    entityRequestData[0]= entityDataRelation;
    VCryptObjectResponse<VCryptObjectResponse<EntityHeader>[]> responseRelation =
vCryptTracker.createOrUpdateEntities(entityRequestData, isReplaceEntity,
```

```
commitBatchSize,requestId);
    assertTrue(responseRelation.isSuccess());
    VCryptObjectResponse<EntityHeader>[] responseArrayRelation=
responseRelation.getObject();
    VCryptObjectResponse<EntityHeader> entityResponseRelation=
responseArrayRelation[0];
    assertTrue(entityResponseRelation.isSuccess());
    EntityHeader entityRelation = entityResponseRelation.getObject();
    Long relationEntityId= entityRelation.getEntityId();
    assertEquals(customerEntityId,relationEntityId);
    Map<String,List<Long>> linkedEntities = entityRelation.getLinkedEntities();
    VCryptDataAccessMgr dataAccessMgr = null;
    VCryptTrackerDataAccess mTrackerDataAccess = null;
    try {
            dataAccessMgr = new VCryptDataAccessMgr();
        mTrackerDataAccess = dataAccessMgr.getVCryptTrackerDataAccess();
    }catch(Exception e) {
            logger.error("Error while creating TrackerEntityFactory instance", e);
    }
    List<VTEntityOneMap> relationships =
mTrackerDataAccess.getVTEntityOneMapByEntityId(customerEntityId,new
Integer(UserDefEnum.getElement(IBharosaConstants.OBJECT_TYPE_ENUM,
"VTEntityDef").getValue()));
    assertEquals(relationships.size(),3);

    // deleting all the relationships for the customer Entity
    Map<String,List<Long>> unlinkEntities = linkedEntities;
    Map<String,String> entityDataMapUnlink = new HashMap<String,String>();
    entityDataMapUnlink.put("$id$",customerEntityId.toString());
    EntityData entityDataUnlink= new
EntityData("customer",entityDataMapUnlink,linkedEntities);
    entityRequestData[0]= entityDataUnlink;
    VCryptObjectResponse<VCryptObjectResponse<EntityHeader>[]> responseDelRel =
vCryptTracker.createOrUpdateEntities(entityRequestData, isReplaceEntity,
commitBatchSize,requestId);
    assertTrue(responseDelRel.isSuccess());
    List<VTEntityOneMap> relationships1=
mTrackerDataAccess.getVTEntityOneMapByEntityId(customerEntityId,new
Integer(UserDefEnum.getElement(IBharosaConstants.OBJECT_TYPE_ENUM,
"VTEntityDef").getValue()));
    assertEquals(relationships1.size(),0);
  }
```

5.2.11 Searching for an Entity on the Basis of Entity ID or Key Data

Use the Search API to perform an entity search based on the Entity Id or values of key entity attributes. The parameters required are the entity name and the entity data map. The entity name corresponds to the entity type and the entity data map contains the entity Id or key value pairs for attributes names and values for the entity to be searched. In case of entity Id, the entity data map is specified as follows:

entityDataMap:

Key	Value
id	107

In this example, `107` is the entity Id of the entity to be searched.

API Call

1. Create a map that contains the entity data of the entity to be searched.

   The map key is the entity attribute name (as specified in OAAM Admin), and the map value is the user input value.

   In the subsequent example, you can see data for the `Customer` entity data, and user input is `Mark Smith` as name. The `Customer` entity has `first name` and `last name` attributes.

   ```
   Map<String,String> entityDataMap = new HashMap<String,String>();
   entityDataMap.put("first name","Mark");
   entityDataMap.put("last name","Smith");
   ```

2. Create an `EntityData` object that encapsulates the entity data map and the entity type. For example, `Customer`.

   ```
   EntityData entityData= new EntityData("Customer",entityDataMap);
   ```

3. Finally, call the API `tracker`. This is a `VCryptTracker` instance.

   ```
   response =tracker.searchEntityByKey(entityData);
   ```

5.3 Data Storage

While creating a transaction instance, the relationship information about the entities involved in the transaction are persisted in the database.

5.3.1 Data Model

`VT_USER_ENTITY1_MAP` stores information about entity relationships. The subsequent table describes the usage of different attributes of `VT_USER_ENTITY1_MAP`.

Attribute	Description
MAP_ID	Primary key for the table, uniquely identifying an entity relationship instance.
ENTITY_ID	Entity ID of the parent entity.
MAP_OBJ_ID	Entity ID of the linked entity.
DEF_MAP_ID	Unique identifier for entity relationship definition.

5.3.2 Metadata

`VT_ENT_DEFS_MAP` stores the entity relationship definitions. The subsequent table describes the usage of different attributes of `VT_ENT_DEFS_MAP`.

Attribute	Description
MAP_ID.	Primary key for the table, uniquely identifying an entity relationship definition.
ENTITY_DEF_ID_1	Entity definition ID for parent entity type.
ENTITY_DEF_ID_2	Entity definition ID for linked entity type.

5.3.3 Expiry of Records

OAAM does not delete expired records. It stores expired unused records in `VT_ENTITY_ONE` and `VT_ENTITY_ONE_PROFILE` tables.

- An entity instance which is stored in the VT_ENTITY_ONE table has an attribute known as expiry time. The expiry time is set to a configurable value while creating an entity instance object. The expiry time changes when an update operation occurs on that entity instance. Whenever the current time exceeds the expiry time, the corresponding record expires.
- The values of attributes for an entity instance are kept in the table, VT_ENTITY_ONE_PROFILE. The profile data also follows the same expiry logic as the entity object in VT_ENTITY_ONE table. However, during an update operation, there is a new record added in the table with new profile data and the previous record expires.

5.3.4 Transaction-Entity Mapping

When an entity is added as an instance to the transaction OAAM stores that association in VT_TRX_ENT_DEFS_MAP. The entity can be extended for entities that are linked to other entities. For example if an entity Customer has an association/link to another entity Address with relation type as Home Address, when the Customer entity is added to a transaction definition then OAAM can store two records into VT_TRX_ENT_MAP, one for the entity Customer and the other for the Address that is referenced by Customer. The inner/nested/child entity Customer.HomeAddress has a reference with the transaction definition with relation_type as Customer$HomeAddress that is derived by concatenating the relation_type between Transaction and Customer entity, dollar($) symbol and the relation_type between Customer entity and Address entity.

5.3.5 Storing Entity Relationships in Transaction Create/Update

While creating a transaction instance, the relationship information about the entities involved in the transaction will be persisted in the database.

When transaction data is created, the following updates occurs:

1. The top-level/directly linked entities are determined.
2. For each top-level/directly linked entities, the following steps are performed:
   a. Determine the nested/chained entities that are associated to this entity by querying the tables VT_ENT_DEFS_MAP (definition associations) and VT_USER_ENTITY1_MAP (data associations)
   b. For each nested/chained entity:

      Look up the transaction association map in VT_TRX_ENT_MAP.

      Get the mapping information using the Transaction Definition Id and Entity Definition Id from VT_TRX_ENT_MAP. Note: This process is similar to an entity instance that is directly associated to the transaction.

      Use the mapping information to create/lookup entity data (VT_ENTITY_ONE, VT_ENTITY_ONE_PROFILE).

      Create a record in VT_ENT_TRX_MAP with the entity Id (from VT_ENTITY_ONE) and transaction Id from the current transaction.

5.4 Common Entity Scenario

Common scenarios are as follows:

- Entities: provider (physician, nurse, and so on), patient and address.

An administrator is investigating a fraud scenario and would like to find out if a provider and patient live or work at the same address. This requires building the following relationships: `patient-address`, `provider-address`; and corresponding rules.

The business rule generates when a patient and physician are working or residing at the same location. The rule returns `true` and generates an alert if `patient.worklocation` = `Physician.worklocation` or `patient.residence` = `Physician.residence`.

- In a medical record access transaction there are only two entities directly involved. That is `Provider ID` and `Patient ID`. No other data is given when calling the API to create the transaction. Both `Providers` and `Patients` have various entities related to them. One example is `home address`, an instance of the `address` entity. Session details shows a medical record access transaction containing a `Provider ID` and a `Patient ID`.

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. In this chapter the Oracle Adaptive Access Manager Proxy for Apache will be referred to as the UIO Apache Proxy.

This chapter provides information on the implementation and use of the UIO Apache Proxy. It contains the following sections:

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

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.

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.

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.

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.

Figure 6–1 shows a Web application before the UIO Proxy is deployed to provide multifactor authentication.

Figure 6–1 Before the Oracle Adaptive Access UIO Proxy

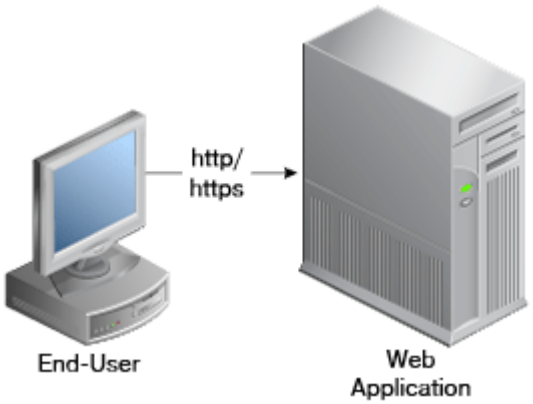
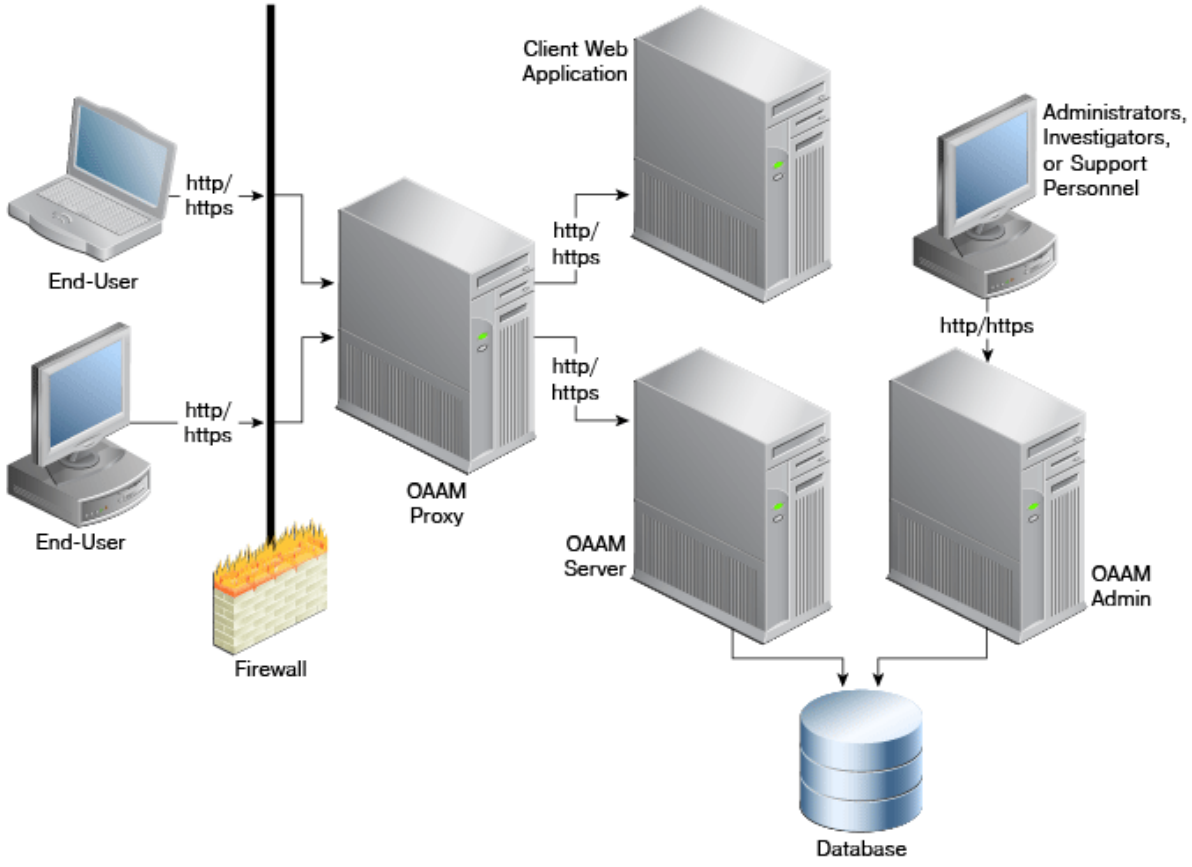


Figure 6–2 shows various components added after the UIO Proxy deployment.

Figure 6–2 After UIO Proxy Deployment



Your local machine running Apache HTTP server. The OAAM sample application is running on an application server on the company intranet. A deployment of OAAM is running on OAAM Server on the company intranet.

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 more information about the Apache HTTP Server, see the Apache HTTP Server Version 2.2 Documentation website at

`http://httpd.apache.org/docs/2.2`

6.2 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 back-end application that has to be protected).

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

Installation Procedure

The installation procedure involves:

- Ensuring that the Apache `httpd` requirements are met

  See Section 6.2.2, "Downloading or Building the Apache httpd."

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

  See Section 6.2.3, "Copying the UIO Apache Proxy and Supported Files to Apache."

- Configuring `memcache` (for Linux only)

  See Section 6.2.4, "Configuring Memcache (for Linux only)."

- Editing the httpd.conf to activate the UIO Proxy

  See Section 6.2.5, "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

  See Section 6.2.6, "Modifying the UIO Apache Proxy Settings."

The post installation procedures involve:

- Setting Up Rules and User Groups

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

- Setting Up Policies

After Installation

After following the installation instructions for OAAM UIO Proxy, the following Apache configuration settings will be set:

- The UIO Module
- The Apache Reverse Proxy Module

- A virtual host entry
- Location mappings for OAAM and the application with ProxyPass and ProxyPassReverse targets for each
- SSL certificate location if required

6.2.1 Before You Begin - UIO Proxy Files for Windows and Linux

For your reference, the UIO Proxy files are summarized in the following tables.

> **Note:** 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 in *$ORACLE_HOME*/oaam/oaam_proxy*platform_specific_file*.

6.2.1.1 Windows

The Windows UIO Proxy binary files are listed in Table 6–1.

Table 6–1 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 Windows UIO Proxy data files are listed in Table 6–2.

Table 6–2 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

6.2.1.2 Linux

The Linux UIO Proxy binary files are listed in Table 6–3.

Table 6–3 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 Linux UIO Proxy data files are listed in Table 6–4.

Table 6–4 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

6.2.2 Downloading or Building the Apache httpd

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

6.2.2.1 Windows

Download the latest Apache `httpd` (2.2.8) build for Windows from the Apache website. Check that the following requirements are met:

- 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

6.2.2.2 Linux

Build the Apache `httpd` following the instructions on the Apache website. When you build Apache, check that the following requirements are met:

- 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

6.2.3 Copying the UIO Apache Proxy and Supported Files to Apache

Copy the UIO Apache Proxy and support files to specific directories in Apache for both Windows and Linux platforms.

6.2.3.1 Windows

Table 6–5 provides information on the following:

- The directories you must copy the UIO Apache Proxy files to after installation
- The tree structure of the UIO Apache Proxy libraries and configuration files, if you installed the files in `C:\Apache2.2`
- The directories the UIO Apache Proxy binary files go into are listed in Table 6–5.

Table 6–5 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.

Move the data files into the directories listed in Table 6–6.

Table 6–6 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

To change the location of the various configuration files, see Section 6.2.5, "Configuring httpd.conf."

6.2.3.2 Linux

After the installation of the Apache `httpd`, perform the following steps:

1. Copy the UIO Apache Proxy binary files into (assuming Apache `httpd` is installed in `/usr/local/apache2`) the directories shown in Table 6–7.

Table 6–7 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
/usr/local/apache2/lib/libapr_ memcache.so.0.0.1	APR Memcache client library.

2. 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
```

3. Ensure that the binary files have executable permission.

4. 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 6–8. (The UIO Apache Proxy data files should follow the directory structure shown in Table 6–8.)

Table 6–8 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

To change the location of the various configuration files, see Section 6.2.5, "Configuring httpd.conf."

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

6.2.4 Configuring Memcache (for Linux only)

This configuration is an optional one 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 the 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 must be centralized. In this case, you can use memcached to store 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 (multithreaded). The MPM is built into the `httpd` and is not a run-time option.

Configure UIO Apache Proxy to Use Memcached for Prefork MPM

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.

Configure Apache httpd to Launch a Single Process for Worker MPM

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 to keep that unchanged, then you must configure UIO Apache Proxy to use memcached. An example of an httpd.conf that you can use to configure a worker MPM to launch a single process is shown below.

```
# 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 6.2.6.1, "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 6.2.6.1, "UIO_ Settings.xml").

Install memcache for Scenarios in which the UIO Apache Proxy Connects to memcache daemon

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` website 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`.

6.2.5 Configuring httpd.conf

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.

6.2.5.1 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:** The setting must be the 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://<QDN_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 steps 4 and 5 are installation-specific and are only examples of the settings you must set. Information on setting details can be found in your Apache documentation.

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_application`

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

6.2.5.2 Configuration with SSL

To enable SSL, refer to the Apache website for Tomcat and your Apache documentation.

The UIO Apache Proxy requires `mod_ssl` to be part of `httpd`. Having the `mod_ssl` as part of `httpd` ensures that the OpenSSL library is linked in and is properly configured for the UIO Apache Proxy to generate session ids. You must 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 website, 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 want 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.

6.2.6 Modifying the UIO Apache Proxy Settings

Modify the UIO Apache Proxy Settings by following the subsequent examples.

6.2.6.1 UIO_Settings.xml

```
<UIO_ProxySettings xmlns="http://example.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="value"/>

        <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 must 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. For information on setting the location for log4j, see your log4j documentation. Log4j is a logging framework (APIs) written in Java.

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 must ensure, on Linux, that the `httpd` process has the permissions to access these files. For information on configuring an application, see Section 6.5, "Configuring the UIO Proxy."

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 6–9.

Table 6–9 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 as needed, 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 through 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 through HTTPS, that will ensure that the cookie is always encrypted when transmitting from client to server, and therefore less likely to be exposed to cookie theft through eavesdropping.
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.

Table 6–9 (Cont.) OAAM UIO Proxy Settings.

Flags	Description
CaptureTraffic	Capture the HTTP traffic - headers and content in the log files. This mode is for debugging purpose. 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. The requirement 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. Expiry time for unused configuration data in memory occurs 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 according to interception phase and prints that out in the logs in microseconds. It is necessary 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

6.2.6.2 UIO_log4j.xml

For actual log4j format details, see your log4j documentation. `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 in Table 6–10.

Table 6–10 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.
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.

Table 6–10 (Cont.) UIO Apache Proxy Log4j Loggers

Loggers	Description
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 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.

6.2.6.3 Application configuration XMLs

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

6.3 Setting Up Rules and User Groups

For information on setting up rules and user groups, see *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*.

6.4 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*.

6.5 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 Apache Proxy uses the XML Relax NG definition which is in the `UIO_Config.rng` file in the proxy distribution.

6.5.1 Elements of the UIO Proxy Configuration File

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

6.5.1.1 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 6–11 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`.

6.5.1.2 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 6–12 lists conditions that can be defined in an interceptor.

Table 6–12 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 6–12 (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 6–12 (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 6–12 (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 reenter 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`. You can use this attribute 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`.

6.5.1.3 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 6–13 lists filters that can be defined in an interceptor.

Table 6–13 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"/>`

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"/>`

Table 6–13 (Cont.) Filters Defined in an Interceptor

Filter name	Attributes	Description
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 6–13 (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	You can use this filter 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 6.5.1.4, "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. For information on using this filter to create a HTTP Basic Authentication header see the example in Section 6.5.1.6, "Filter Examples - FormatString." FormatString is not supported in the UIO Apache Proxy. As it ProcessString provides all the required functionality.

6.5.1.4 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 prior 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

6.5.1.5 ProcessString Encoding/Decoding Schemes for Special Characters URL Encoded in OAAM Change Password

When using OAAM UIO Proxy to call OAAM change password if the password value contains special characters they are URL encoded when passed to OAAM.

Four encoding/decoding schemes are provided for the ProcessString filter so that the UIO Proxy can use URL-encoding or the same encoding schemes that the OAAM Server uses.

ProcessString Encoding/Decoding Schemes

The encoding/decoding schemes that can be used in the ProcessString filter in the OAAM UIO Proxy are:

Table 6–14 ProcessString Encoding/Decoding Schemes

Encoding/Decoding Schemes	Details
asadecode:	Decodes the OAAM Server encoded strings
asaencode	Encodes the string value using the same schema that the OAAM server uses
urldecode	Decodes the string value that is URL-encoded
urlencode	Performs URL encoding for the given string

Example

The 11g OAAM Server performs UTF8 encoding for the credentials post to it. In response headers of oaam_server/changePassword.do, headers of newPassword and confirmPassword will be encoded. In this case, if the protected application does not accept such encoded credentials, the following interceptor could be used as an example to decode the encoded values of those credentials and save them to UIOProxy variables. The UIO Proxy can then perform actions with these variables, such as post their values to the protected application.

Sample Interceptor

```
<ResponseInterceptor id="EncodingDecodingSchemes"
```

```
desc="ProcessString eval with encoding/decoding tags" enabled="true">
 <ResponseUrl url="/oaam_server/changePassword.do"/>
   <Conditions>
   <VariableValue name="%REQUEST_METHOD" value="POST"/>
    <HeaderPresent  name="password:"/>
    <HeaderPresent  name="newpassword:"/>
    <HeaderPresent  name="confirmpassword:"/>
   </Conditions>
   <Filters>
    <SaveHeader name="newpassword:" variable="%newpassword"/>
    <ProcessString source="[%newpassword]" variable="%newpassword1"
         action="eval" find="sub-string" encoding="asadecode"/>
    <AddHeader name="newpasswordASADecoded:" value="%newpassword1"/>
    <ProcessString source="[%newpassword1]" variable="%newpassword2"
          action="eval" find="sub-string" encoding="asaencode"/>
    <AddHeader name="newpasswordASAEncoded:" value="%newpassword2"/>
     <ProcessString source="[%newpassword1]" variable="%newpassword3"
         action="eval" find="sub-string" encoding="urlencode"/>
    <AddHeader name="newpasswordUrlEncoded:" value="%newpassword3"/>
     <ProcessString source="[%newpassword3]" variable="%newpassword4"
          action="eval" find="sub-string" encoding="urldecode"/>
    <AddHeader name="newpasswordURLDecoded:" value="%newpassword4"/>
    </Filters>
 </ResponseInterceptor>
```

6.5.1.6 Filter Examples - FormatString

An example is presented below on how to create an HTTP Basic Authentication response header in variable `$AuthHeader`Value, using the user name and 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"/>
```

6.5.1.7 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 must 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 must 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.

6.5.1.8 Variables

The proxy variables can store string data in the proxy memory. You can use variables in conditions, filters and actions. For example, you can use the `SaveHeader` filter to save the value of 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. You can use the `ClearSession` filter 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 must be preserved across requests and across clients.

Global variables can be set at the proxy configuration load time using `SetGlobal` in the configuration file.

Pre-defined variables

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

Table 6–15 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. For the 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.

Table 6–15 (Cont.) Pre-defined Variables Supported by the UIO Proxy

Variable name	Description
%REMOTE_HOST	Host name of the client or agent of the client. (For the UIO Apache Proxy, you must enable the host name lookup by using the Apache directive '`HostnameLookups On`'.)
%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

6.5.1.9 Application

You can use a single proxy installation 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 before the interceptors defined in the applications.

6.5.2 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. 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. For example, 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.

6.5.3 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 protocol uses HTTP headers to exchange data between the UIO Proxy and OAAM Server. Table 6–16 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 6–16 OAAM Server Interface

URL	Condition	Action
Any request to OAAM Server page	On receiving request	Set header "`BharosaAppId`". OAAM Server will use this header value to select appropriate customizations (UI, rules, 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 6–16 (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 6–16 (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 through 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, and others.)
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 6–16 (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>"

6.6 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 through 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 through the proxy. The captured HTTP traffic should then be analyzed to author the proxy configuration.

6.6.1 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, you can use the live application.

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. During the discovery process some accounts could become disabled, due to multiple invalid login attempts.

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

6.6.2 Setting Up the UIO Apache Proxy

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

Table 6–17 shows the settings (in `UIO_Settings.xml`) to enable this mode of operation.

Table 6–17 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 file name before the start of the scenario.

After application discovery is performed, the proxy settings should be set, as shown in Table 6–18, to restore the default UIO Apache Proxy behavior.

Table 6–18 Settings to restore default proxy behavior

Settings	Value
IgnoreUrlMappings	0
CaptureTraffic	0

6.6.3 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 must be studied carefully to derive this information. A few ways applications respond on successful login are the following:
   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.

6.7 OAAM Sample Application

The proxy configuration to add multifactor authentication to the BigBank Web application is shown in this section. The BigBank web application is an OAAM 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 Apache proxy use:

```
<?xml version="1.0" encoding="utf-8"?>
<BharosaProxyConfig xmlns="http://example.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, you use BigBank's login form to collect username and
           password -->
    <!-- In phase two, you 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 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
                           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 OAAM Server 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 OAAM Server 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 is not 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 OAAM 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 OAAM 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="OAAM 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 OAAM 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 OAAM proxy session">
  <ResponseUrl url="/bigbank/logout.do"/>

  <Filters>
    <ClearSession/>
  </Filters>

  <!-- Target action="redirect-client" url="/oaam_server/loginPage.jsp"/ -->
```

```
        </ResponseInterceptor>

    </Application>
</BharosaProxyConfig>
```

6.7.1 Descriptions for Interceptors

Descriptions of the various interceptors that are defined in the sample configuration are summarized in Table 6–19.

Table 6–19 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 login page. 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 the OAAM 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`.

Table 6–19 (Cont.) Sample Configuration Interceptors

Interceptor ID	Type	Explanation
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.
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.

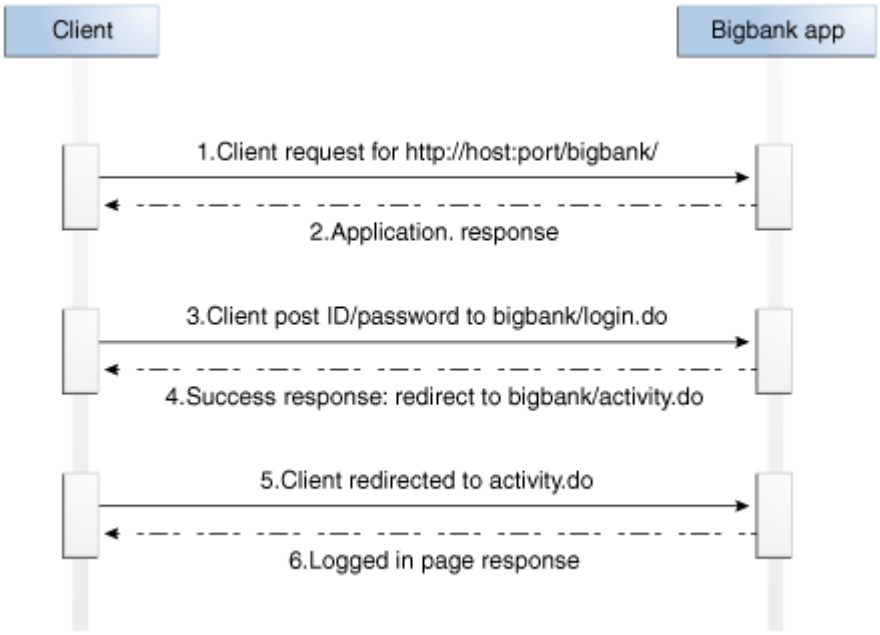
6.7.2 Flow for BigBank without UIO Proxy

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

6.7.2.1 Login

Figure 6–3 shows the Login without UIO Proxy flow.

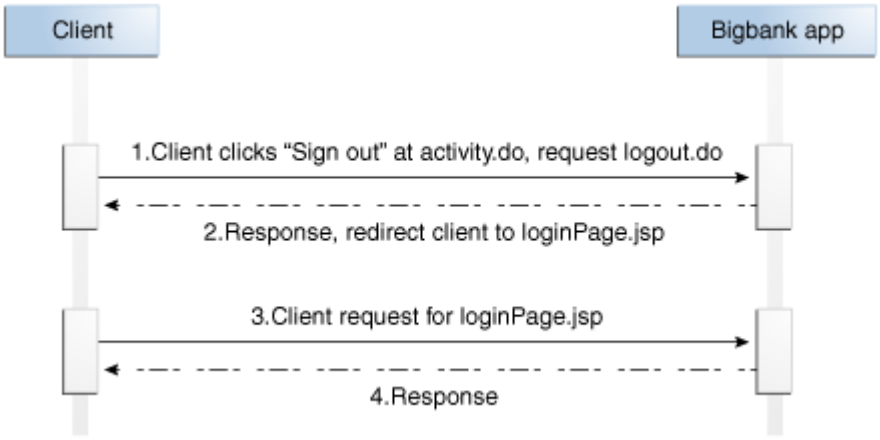
Figure 6–3 Login Flow - Without UIO Proxy



6.7.2.2 Logout

Figure 6–4 shows the Logout without UIO Proxy flow.

Figure 6–4 Logout - Without UIO Proxy



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

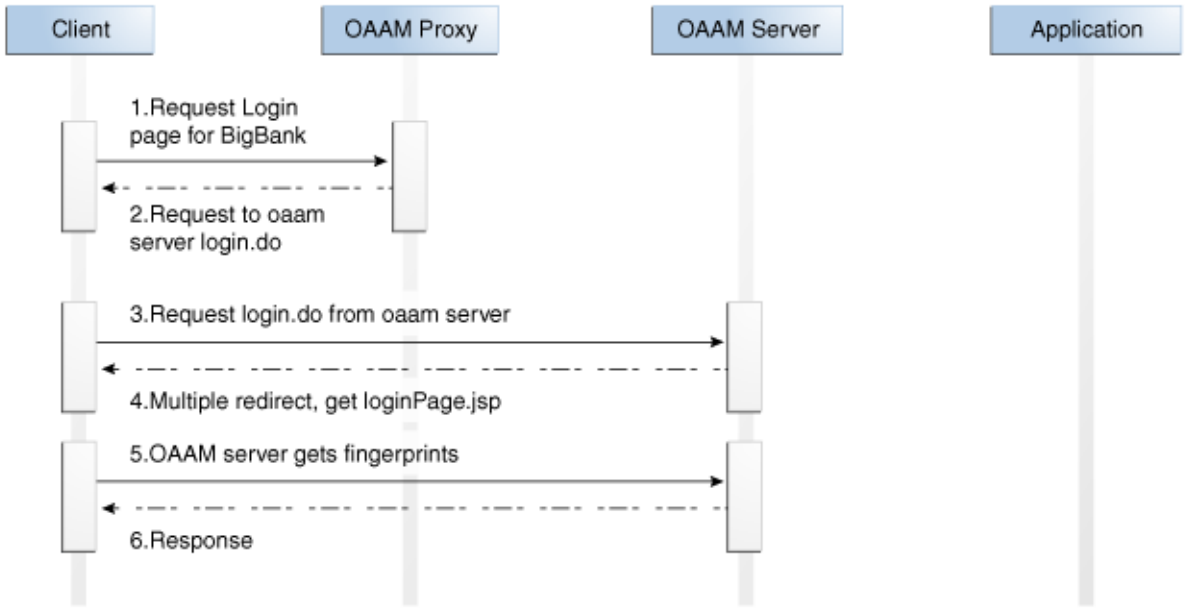
This section provides details for the flows for first time users who log in to the BigBank application through the UIO Proxy. The regular flow, including the login phase, registration phase/skip registration phase, and logout phase, and the deviation flow (block login) are covered. Interceptors defined 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 when the interceptors match request/response. Normal messages that the proxy passes between the 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.

Figure 6–5 shows the flow for Getting the Login Page (Login phase).

Figure 6–5 Flow for Getting Login Page



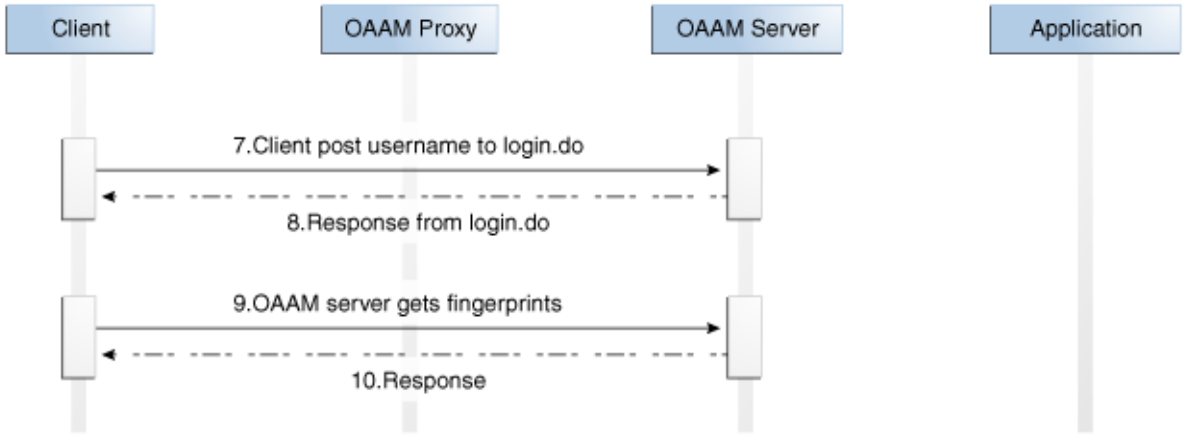
1. Client requests Login page for the application (`http://proxyhost:port/bigbank`).
2. The proxy intercepts the request, and sets the headers. Then, the proxy redirects the client to `oaam_server/login.do`.

   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, as shown in Figure 6–6.

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

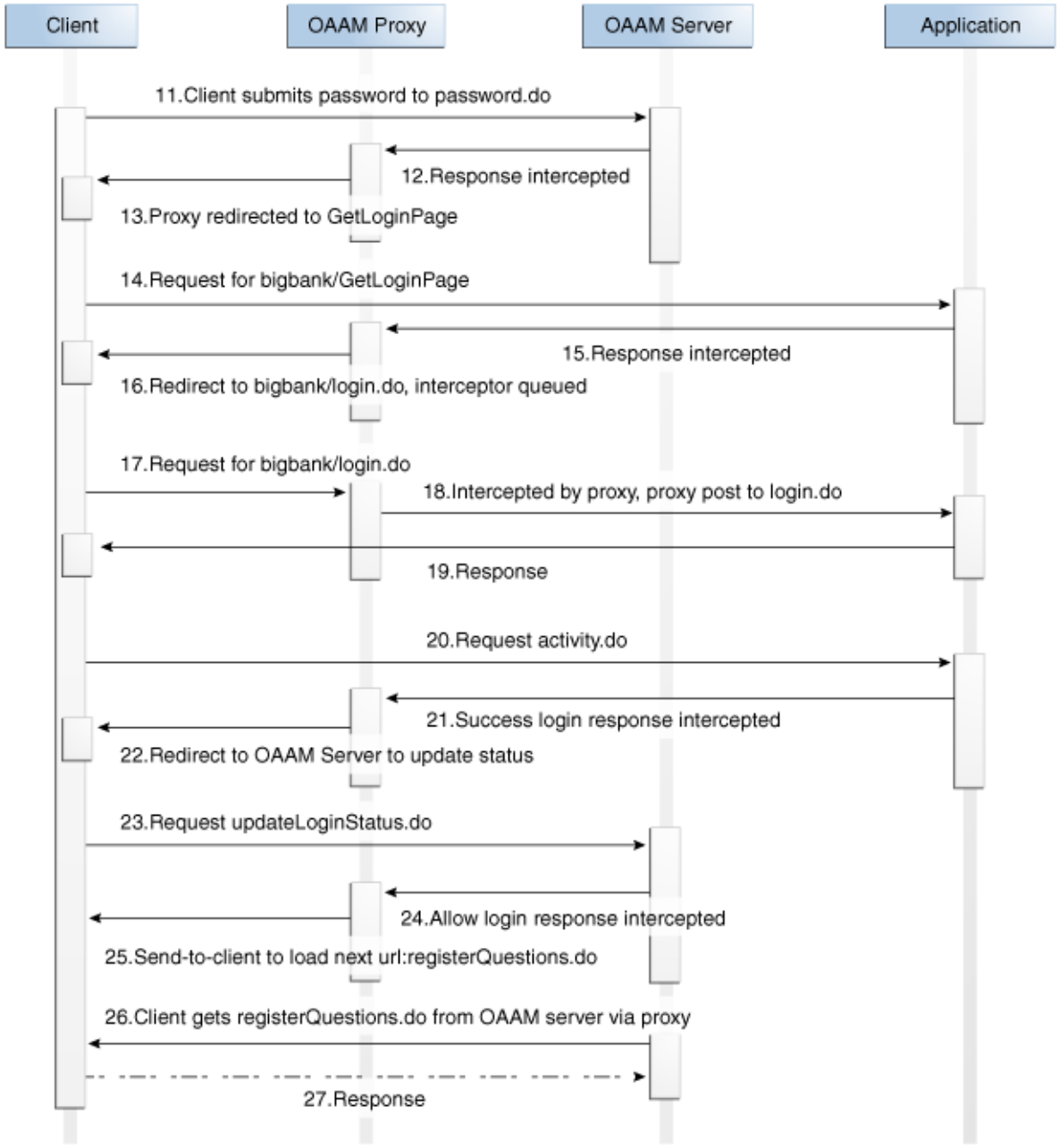


7. The client posts the user name to the OAAM Server (`http://proxyhost:port/oaam_server/login.do`).

   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, as shown in Figure 6–7.

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`.

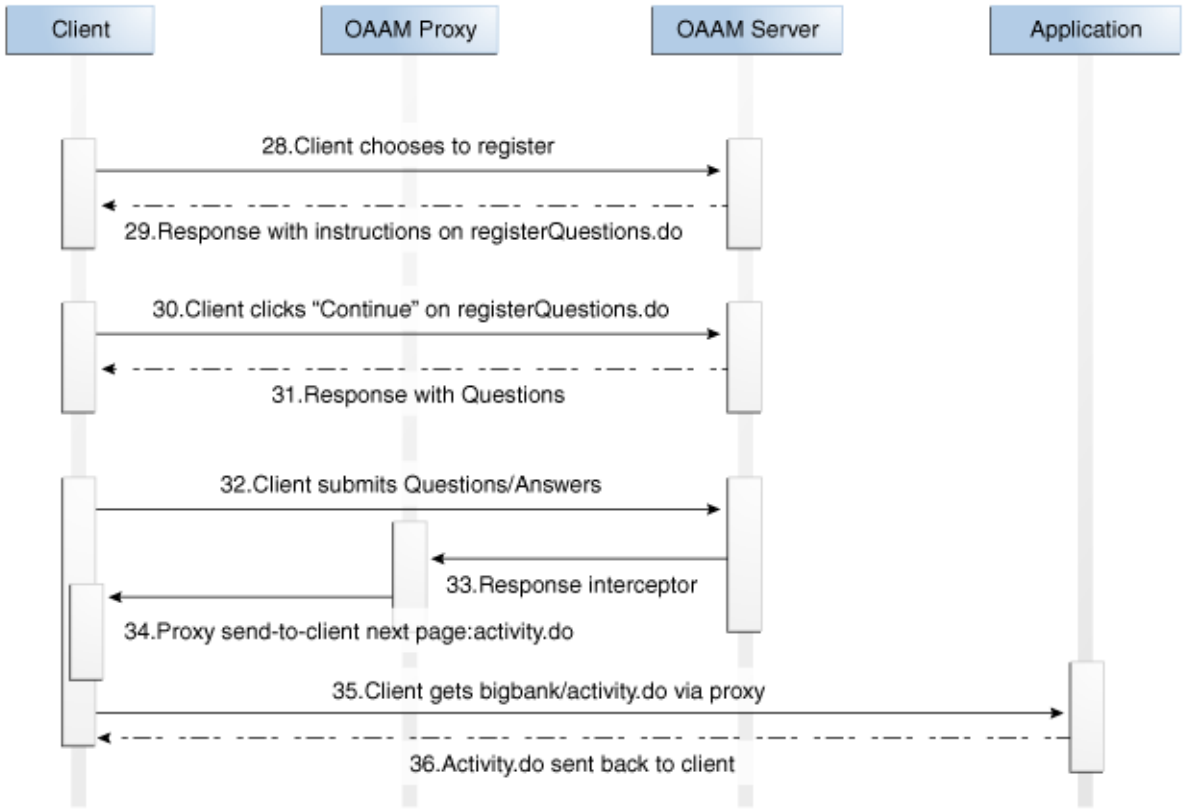
25. The proxy takes the `send-to-client` action. It sets the `display-url` variable so that the client will request this URL after receiving the response.

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.

    Then registration flow where the client chooses to register is shown in Figure 6–8.

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

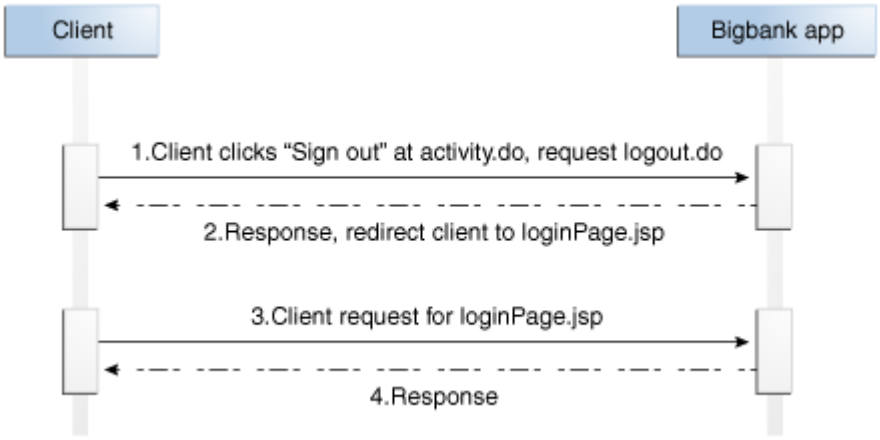


28. The client chooses to register (`Post` to `http://proxyhost:port/oaam_server/registerQuestions.do`).
29. OAAM Server responds with instructions.
30. The client clicks **Continue** on the instruction page. (`http://proxyhost:port/oaam_server/registerQuestions.do`).
31. OAAM Server responds with the 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.

    The Logout Phase is shown in Figure 6–9.

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

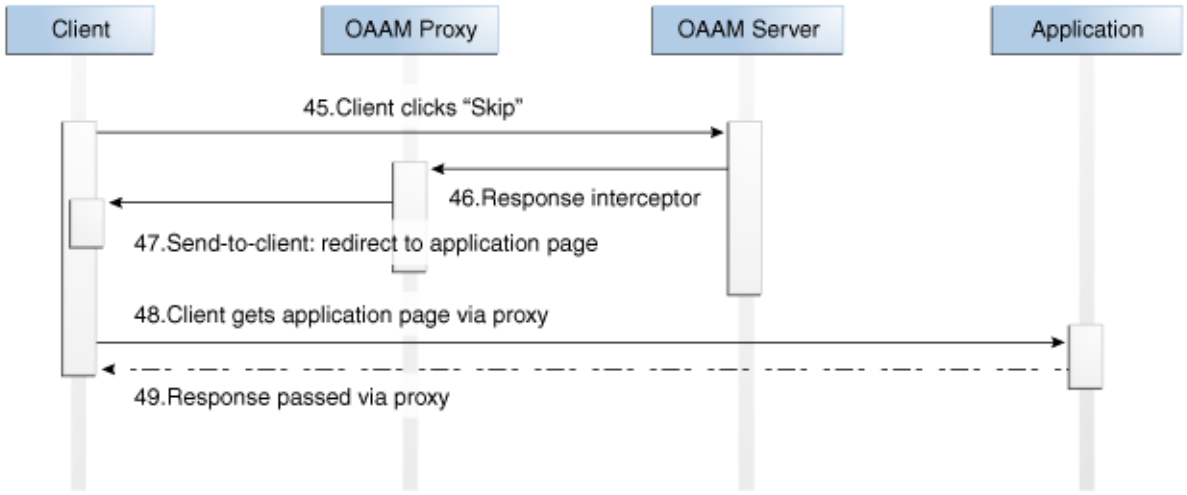


37. The client clicks **Log out** (`http://proxyhost:port/bigbank/logout.do`).
38. The application sends back a response and redirects the client to `bigbank/loginPage.jsp`.

    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.

    Figure 6–10 shows the Skip Registration phase where the client chooses to skip the registration of questions. This phase occurs after the Login phase in regular flow.

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

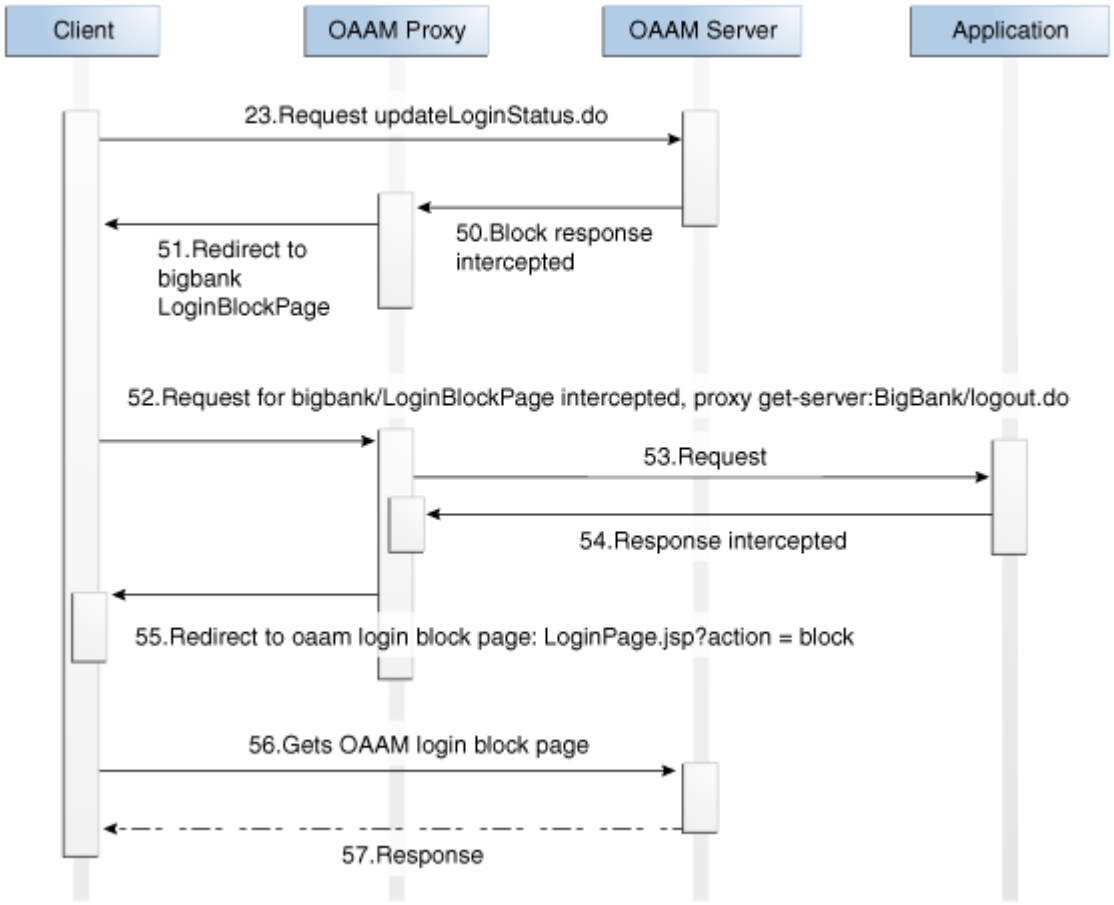


45. The client chooses to skip the registration (Post to `http://proxyhost:port/oaam_server/registerQuestions.do`).
46. OAAM Server responds.
47. The proxy intercepts the response and redirects the client.

    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.

    Figure 6–11 shows the Deviation flow - Block login, which occurs when OAAM Server decides to block the client after the 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.

6.8 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.

6.8.1 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.

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 must 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

6.8.2 Patch Unsuccessful

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

Part III

Customization and Extensions

Part III contains information about customizations and extensions.

It contains the following chapters:

7

Using the OAAM Extensions Shared Library to Customize OAAM

Shared libraries are collections of programming and data that multiple applications can use. They can permit applications to use memory efficiently by sharing common programming and resources.

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 Customizing or Extending OAAM By Editing Enums

To override any Oracle Adaptive Access Manager properties or extend OAAM enumerations, add those properties and enumerations to `oaam_custom.properties`. Later, you will save that file in the `oaam_extensions/WEB-INF/classes/bharosa_properties` folder.

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
```

7.2 Adding Customizations Using the OAAM Extensions Shared Library

You can customize Oracle Adaptive Access Manager by adding custom JAR and JSP files and other files to the OAAM Extensions Shared Library and editing property files.

The OAAM Extensions 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 servers. By default `oracle.oaam.extensions.war` contains the `MANIFEST.MF`, which has the definition of the OAAM Extensions Shared Library.

This section provides instructions for adding customizations to Oracle Adaptive Access Manager.

7.2.1 Prerequisite

Ensure the property `bharosa.uio.proxy.mode.flag` is set as appropriate.

The default for the property `bharosa.uio.proxy.mode.flag` is `false`. If you are using an UIO proxy deployment, the property should be set to `true`. To configure custom branding for multitenancy with the OAAM Proxy, the property `bharosa.uio.proxy.mode.flag` must be set to `true`.

If you are adding customizations and also configuring integration with Oracle Access Management Access Manager 11*g* using the TAP scheme, the property must be set as `false`. Setting the property to `true` causes OAAM and Access Manager integration using TAP to fail with the following message:

```
Sorry, the identification you entered was not recognized.
```

7.2.2 Step 1 Extract the OAAM Extensions Shared Library

To extract the OAAM Extensions Shared Library, proceed as follows:

1. Create a work folder named `oaam_extensions`.

   The folder can be created anywhere if it is outside the installation folder.

2. Extract the `oracle.oaam.extensions.war` file into the work folder.

   In the `oaam_extensions` folder, you should see the following subfolders:

   - `META-INF`
   - `WEB-INF`
   - `WEB-INF\lib`
   - `WEB-INF\classes`

7.2.3 Step 2 Create a MANIFEST.MF File

In the `META-INF` folder, create a file named `MANIFEST.MF` and ensure it contains the following lines:

```
Extension-Name: oracle.oaam.extensions
Implementation-Version:99.9.9.9.9
Specification-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`. Errors are thrown if the version is not incremented.

7.2.4 Step 3 Compile Custom Java Classes

Compile custom java classes that extend or implement Oracle Adaptive Access Manager classes, adding the JAR files from the *ORACLE_IDM_HOME*`\oaam\cli\lib` folder to the build class path.

7.2.5 Step 4 Add Custom JARs and Files

Add the custom jars and files as described:

1. Add the custom JAR files to the *IAM_Home*`\oaam\oaam_extensions\generic\WEB-INF\lib` folder.
2. Make changes to `oaam_custom.properties` and save it to the `oaam_extensions\WEB-INF\classes\bharosa_properties` folder.
3. Add custom JSP files directly to `oaam_extensions`.

7.2.6 Step 5 Repackage the OAAM Extensions Shared Library

Repackage the OAAM Extensions Shared Library, `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/ .
```

7.2.7 Step 6 Verify If the Repackaged WAR File Contains the Custom JAR Files

Verify that the repackaged WAR file contains the custom JAR files. Use the following command to view its contents:

```
jar tvf oracle.oaam.extensions.war
```

7.2.8 Step 7 Stop All Managed Servers

Stop all managed servers if they are running:

```
MW_HOME/user_projects/domains/domain_name/bin/stopManagedWeblogic.sh
  oaam_admin_server1
MW_HOME/user_projects/domains/domain_name/bin/stopManagedWeblogic.sh
  oaam_server_server1
```

7.2.9 Step 8 Start the WebLogic Administration Server

Start the WebLogic Administration Server:

```
MW_HOME/user_projects/domains/domain_name/bin/startWeblogic.sh
```

7.2.10 Step 9 Log In to the WebLogic Administration Console

Start the WebLogic Server where Oracle Adaptive Access Manager is deployed and log in to the WebLogic Administration Console:

```
http://hostname:port/console
```

7.2.11 Step 10 Deploy the New OAAM Extensions Shared Library

Deploy the new `oracle.oaam.extensions.war` file as a shared library with `oaam_server_server1` and `oaam_admin_server1` as target applications.

1. Navigate to Domain **Environment** > **Deployments** and lock the console.
2. Click the **Install** button.
3. Browse to the location of the `oracle.oaam.extensions.war` file and select it by clicking the option next to the WAR file and clicking **Next**.
4. Ensure **Install this deployment as a library** is selected and click **Next**.
5. Select deployment targets, `oaam_admin_server1` and `oaam_server_server1`.
6. Click **Next** again to accept the defaults in this next page and then click **Finish**.
7. Click the **Save** button and then A**ctivate Changes**.
8. Start the OAAM Admin and OAAM managed servers.

```
MW_HOME/user_projects/domains/domain_name/bin/startManagedWeblogic.sh
  oaam_admin_server1
MW_HOME/user_projects/domains/domain_name/bin/startManagedWeblogic.sh
  oaam_server_server1
```

7.2.12 Step 11 Test the Functionality

Test the custom functionality and make sure files added to `oracle.oaam.extensions.war` are used by Oracle Adaptive Access Manager applications.

8

Customizing OAAM Web Application Pages

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 Tips for Customizing the OAAM Web Application Pages

As you plan to customize the web user interface, keep the following points in mind:

- When customizing, you often copy files that are installed with OAAM into a directory in which you can modify them. By modifying files in this directory, you prevent your modifications from being overwritten when the software is upgraded.
- When configuring the web application, use `oaam_custom.properties`. The file should contain:
  - Client-configured properties (any properties that have been customized for a specific deployment)
  - 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.

  In the deployed application, the `oaam_custom.properties` file is located in the `web-inf/classes` directory.

  Note: In 11.1.2, the `oaam_custom.properties` file replaces the bharosa_server.properties file from previous versions.

- When adapting the OAAM deployment to a particular language, use `client_resource_`*`locale`*`.properties` where *`locale`* is the locale string for which you

want to use the custom values (en, es, and others). The client_resource_*locale*.properties file is used to customize text on the pages when the application is translated into many languages. The file should contain

- Client-configured properties that are configurable for each locale being supported. *locale* is the locale string for which you want to use the custom values (en, es, and others).
- 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.

Note: The client_resource_*locale*.properties file is not used for header and footer files customization.

The administrator creates the client_resource_*locale*.properties to customize the application so that it contains locale-specific properties.

For instructions on customizing, extending, or overriding Oracle Adaptive Access Manager properties, see Chapter 7, "Using the OAAM Extensions Shared Library to Customize OAAM."

8.2 OAAM Properties

You can manage the appearance and behavior of OAAM using user-defined enumerations. User-defined enumerations 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.

8.2.1 Enum Example

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.

8.2.2 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 8.2.1, "Enum Example," 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, see Chapter 7, "Using the OAAM Extensions Shared Library to Customize OAAM."

8.2.3 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 8.2.2, "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
```

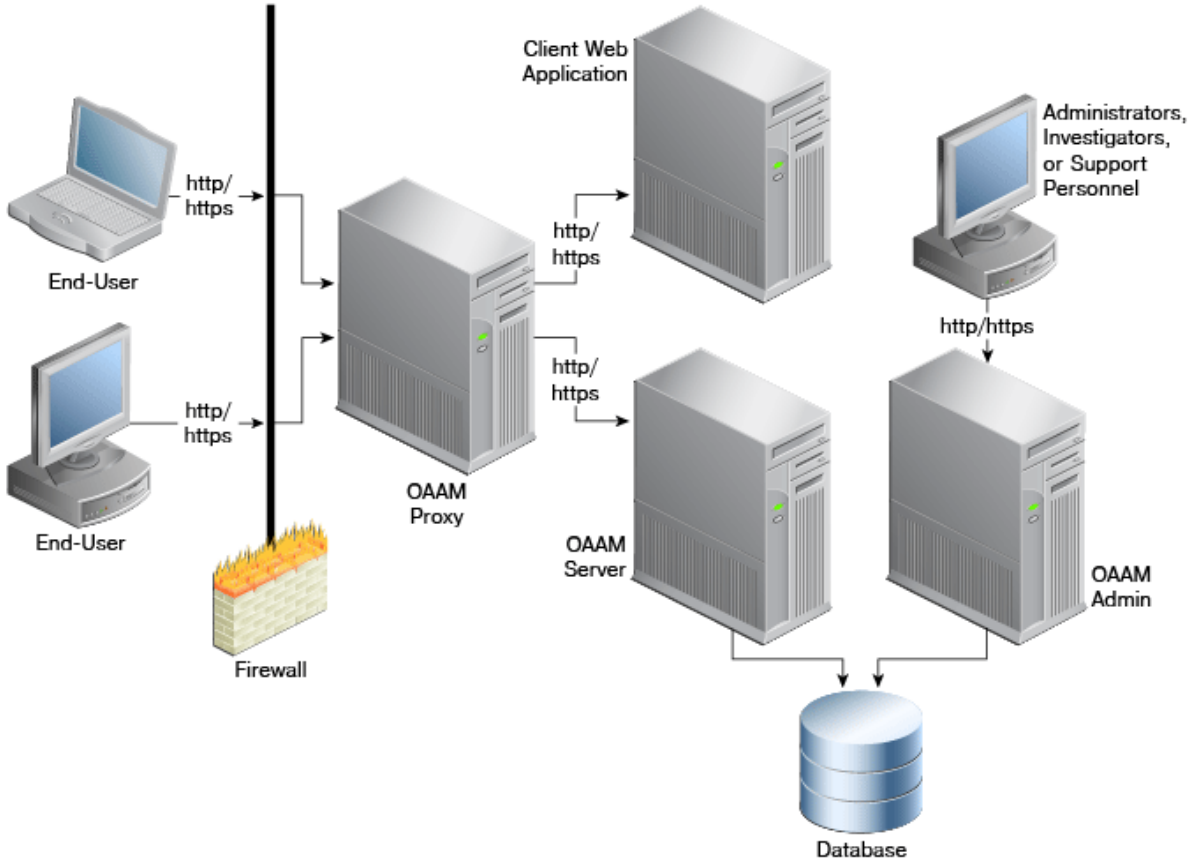
8.3 Customizing the OAAM Server for Multiple Applications

Multitenancy refers to a principle in software architecture where a single instance of the software runs on a server, serving multiple client organizations. With a multitenant architecture, each client organization feels as if they are working with a separate customized application instance.

You can configure the OAAM Server to support one or more Web application authentication and user registration flows. The OAAM Server configuration is specific to the UIO Proxy deployment. The OAAM UIO Proxy offers multifactor authentication to Web applications without requiring any change to the application code.

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 determines the risk and takes the appropriate actions, such as permitting the login, challenging the user, blocking the user, and other actions.

Figure 8–1 Universal Installation Deployment



The UIO Proxy can be placed in front of multiple applications and customized to work with each one as required.

To ensure that a customer's data is unique from that of other customers, an Application ID for the client application is mapped to an Organization ID. An Organization ID identifies what tenant applications a user utilizes.

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

Determine how many applications are to be configured and 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.

This Application ID is the same one used to configure the Proxy (see Chapter 6, "Oracle Adaptive Access Manager Proxy").

The Proxy will send the `AppId` to OAAM Server as needed through 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.

Properties can contain an `AppId` to allow for multiple configurations for multi-tenant scenarios.

The following example shows how to use the `AppId` to define a property in the OAAM server:

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

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

8.3.2 Determining Default User Groups

You can configure each application 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 the user is first created in the Oracle Adaptive Access Manager database. The Organization ID is used when a user attempts to log in to the application and user data is loaded from the database.

An example of how Organization ID is used in a property definition is shown 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`.

8.3.3 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.

Property definitions that start with `bharosa.uio.default` apply to all Application IDs unless overridden by a more specific value.

In the following example, `default` is used instead of the `appId` to designate the property as a global default. The property is used across all applications of the OAAM Server installation unless a specific application has another location specified.

```
bharosa.uio.default.header = /globalcustomHeader.jsp
bharosa.uio.default.footer = /globalcustomFooter.jsp
```

The default properties for the path to the custom header and footer are:

```
bharosa.uio.default.header = path_to_custom_header.jsp
bharosa.uio.default.footer = path_to_custom_footer.jsp
```

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

In Oracle Adaptive Access Manager and Access Manager integrations, the `AppId` is `oam`. This allows OAAM to display a different header and footer that keeps the same look and feel as Access Manager pages.

The properties for the path to the Access Management custom header and footer are:

```
bharosa.uio.oam.header = path_to_custom_header.jsp
bharosa.uio.oam.footer = path_to_custom_footer.jsp
```

These specific prefix `bharosa.uio.oam` value overrides the default settings defined as `bharosa.uio.default`.

In the following 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.

```
bharosa.uio.default.header = /globalcustomHeader.jsp
bharosa.uio.default.footer = /globalcustomFooter.jsp
bharosa.uio.app1.header = /app1customHeader.jsp
bharosa.uio.app1.footer = /app1customFooter.jsp
bharosa.uio.app2.footer = /app2customFooter.jsp
```

8.3.4 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/customFooter.jsp

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

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

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

8.4 Customizing the Appearance of OAAM Server Pages

This section describes how to customize the appearance of the OAAM server pages.

The 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 Customizing Headers and Footers

You can create custom header and footer files for the applications being secured. The header and footer files are JSP files and can contain any HTML or JSP code required to replicate the look of the application being secured.

1. Create a work folder called `oaam_extensions`. (The folder can be created anywhere if it is outside the installation folder.)
2. Locate `oracle.oaam.extensions.war` in the following directory:

   IAM_Home`/oaam/oaam_extensions/generic`

3. Extract `oracle.oaam.extensions.war` in the `oaam_extensions` folder.
4. In the `oaam_extensions` folder, create the following subfolders:

   ```
   /client/app1/
   /client/app1/images/
   ```

5. Create a `customHeader.jsp` and `customFooter.jsp` inside the `client/app1/` folder.

   The header (`customHeader.jsp`) and footer (`customFooter.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 `customHeader.jsp` for the header.

   ```
   /client/app1/customHeader.jsp
        <img src="client/app1/images/customHeader.jpg" alt="Welcome to App1"/>
   ```

   Copy the following code into customFooter.jsp for the footer.

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

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

6. Add associated files to the `client/app1` folder as needed.

   For example, the `customHeader.jpg` and `customFooter.jpg` image files referenced by `customHeader.jsp` and `customFooter.jsp`.

   ```
   /client/app1/images/customHeader.jpg
   /client/app1/images/customFooter.jpg
   ```

7. Open the `oaam_custom.properties` file in the `WEB-INF/classes/bharosa_properties` directory of the `oracle.oaam.extensions.war` file.
8. To associate these header and footer files with the application, add the following properties to `oaam_custom.properties` and save it to *`IAM_HOME`*`\oaam\oaam_extensions/WEB-INF/classes/bharosa_properties`.

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

9. Repackage `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/ .
   ```

   Note: Note that there is a dot at the end of the command.

   This command recreates the WAR file with the `MANIFEST.MF` file. The new JSP files, referenced images, and added properties in `oaam_custom.properties` are included in the new WAR file.

10. Shut down the OAAM Admin and OAAM Server managed servers.

11. Start the WebLogic Server where Oracle Adaptive Access Manager is deployed and log in to the WebLogic Administration Console.
12. Navigate to **Domain Environment** > **Deployments** and lock the console.
13. Click the **Install** button.
14. Browse to the location of the `oracle.oaam.extensions.war` file and select it by clicking the radio button next to the `.war` file and clicking **Next**.
15. Ensure **Install this deployment as a library** is selected and click **Next**.
16. Select OAAM Admin and OAAM Server servers as deployment targets.
17. Click **Next** again to accept the defaults in this next page and then click **Finish**.
18. Click the **Save** button and then **Activate Changes**.
19. Start the OAAM Admin and OAAM Server managed servers.

8.4.2 Customizing Content and Messaging

You can customize content and messaging of the OAAM server pages by adding properties to the `client_resource_`*`locale`*`.properties` file.

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 customize content and messaging:

1. Create a work folder called `oaam_extensions`. (The folder can be created anywhere if it is outside the installation folder.)
2. Locate `oracle.oaam.extensions.war`, which is located in the *`IAM_Home`*`/oaam/oaam_extensions/generic` directory.
3. Explode `oracle.oaam.extensions.war` into the `oaam_extensions` folder.
4. Create a `client_resource_`*`locale`*`.properties` in *`IAM_Home\oaam\oaam_extensions\generic\WEB-INF\classes`*.
5. Add the customized message to this file.

   For example, to change the page title on the login page for the `appId1` application, add the following line to `client_resource_`*`locale`*`.properties`:

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

   For example, to customize the error message displayed when a user has been blocked by security rules, add the following line to `client_resource_`*`locale`*`.properties`:

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

6. Repackage `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. Shut down all managed servers.
8. Start the WebLogic Server where Oracle Adaptive Access Manager is deployed and log in to the Oracle WebLogic Administration Console.

9. Navigate to Domain **Environment** > **Deployments** and lock the console.
10. Click the **Install** button.
11. Browse to the location of the `oracle.oaam.extensions.war` file and select it by clicking the option next to the WAR file and clicking **Next**.
12. Ensure **Install this deployment as a library** is selected and click **Next**.
13. Select OAAM Admin and OAAM Server servers as deployment targets.
14. Click **Next** again to accept the defaults in this next page and then click **Finish**.
15. Click the **Save** button and then A**ctivate Changes**.
16. Start the OAAM Admin and OAAM Server managed servers.

8.4.3 Modifying User Interface Styles

You can create a custom Cascading Style Sheet (CSS) to create a custom user interface. The CSS file provides control over backgrounds, font colors and sizes, and so on. The default CSS file, `oaam_uio.css`, is located in the `css` folder. You can override the styles in this CSS file using a custom CSS file. Use the file for an application or at a global level. For information on setting application properties, see Section 8.3.3, "Configuring Application Properties."

For example, to override the font-family of the default body style definition:

1. Create a work folder called `oaam_extensions`.

   The folder can be created anywhere if it is outside the installation folder.
2. Locate `oracle.oaam.extensions.war`, which is located in the *`IAM_Home`*`/oaam/oaam_extensions/generic` directory.
3. Explode `oracle.oaam.extensions.war` into the `oaam_extensions` folder.
4. Create the `client/app1/css` directory.
5. Create an `app1.css` file.
6. Add the following code to the `app1.css` file.

```
body{
    background-color:#ffffff;
    font-size:12px;
    color:#000000;
    font-family:arial,helvetica,sans-serif;
    margin:0px 0px 0px 0px
}
```

7. Change Helvetica to the primary font-family you want to use for your `appId1` application.
8. Add the file to the `/client/app1/css` directory.
9. Open the `oaam_custom.properties` file in the `WEB-INF/classes/bharosa_properties` directory of the `oracle.oaam.extensions.war` file.
10. To use the newly created file, set the following property in `oaam_custom.properties`:

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

11. Repackage `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/ .
```

12. Shut down the OAAM Admin and OAAM Server managed servers.
13. Start the WebLogic Server where Oracle Adaptive Access Manager is deployed and log in to the Oracle WebLogic Administration Console.
14. Navigate to **Domain Environment** > **Deployments** and lock the console.
15. Click the **Install** button.
16. Browse to the location of the `oracle.oaam.extensions.war` file and select it by clicking the option next to the WAR file and clicking **Next**.
17. Ensure **Install this deployment as a library** is selected and click **Next**.
18. Select OAAM Admin and OAAM Server servers as deployment targets.
19. Click **Next** again to accept the defaults in this next page and then click **Finish**.
20. Click the **Save** button and then **Activate Changes**.
21. Start the OAAM Admin and OAAM Server managed servers.

Any style defined in the `oaam_uio.css` in the OAAM Server ear file can be overridden in this manner if required.

8.5 Enabling the Single Login Page

To use a single login page, you must:

- Configure the OAAM login page so that the **Where is my password** link does not display
- Configure the OAAM login page to accept the password along with the user name
- Enable the password field on the OAAM login page
- Ensure that OAAM is configured to use OAAM HTML Pad instead of the virtual authentication devices

The properties documented in the following sections will be modified in `oaam_custom.properties` and the changes deployed as part of the OAAM Shared Extensions Library:

- Section 8.5.1, "Configuring the OAAM Single Login Page So That the "Where is my password" Link Does Not Display"
- Section 8.5.2, "Configuring the OAAM Single Login Page to Accept the Password Along with the User Name"
- Section 8.5.3, "Enabling the Password Field in the OAAM Single Login Page"

> **Note:** In cases where the property does not exist in the `oaam_custom.properties` file, you will have to add it.

For instructions on deploying the OAAM Shared Extensions Library, see Chapter 7, "Using the OAAM Extensions Shared Library to Customize OAAM."

8.5.1 Configuring the OAAM Single Login Page So That the "Where is my password" Link Does Not Display

Set the following property to `false` so that the **Where is my password?** link is not displayed:

`bharosa.uio.default.signon.links.enum.wherepassword.enabled`

8.5.2 Configuring the OAAM Single Login Page to Accept the Password Along with the User Name

To configure OAAM server to accept both user name and password from the OAAM login page, set the following to `true`:

`bharosa.uio.default.login.auth.enabled`

8.5.3 Enabling the Password Field in the OAAM Single Login Page

To enable the Password field on the login page, set the following to `true`:

`bharosa.uio.default.credentials.enum.password.enabled`

Then, restart the OAAM Server.

8.5.4 Ensuring that OAAM is Configured to Use the OAAM HTML Pad Instead of the Virtual Authentication Devices

Ensure that you have configured OAAM so that virtual authentication devices are not used in any flow. The two ways to configure OAAM to use the OAAM HTML Pad are documented in this section.

Configure the OAAM AuthenticationPad Policy to Use the OAAM HTML Pad for Challenge SMS, Registered Image and Caption, Challenge Email, and Challenge Question Rules

1. Log in to the OAAM Administration Console as an administrator.
2. Double-click the **Policies** node. The **Policies Search** page is displayed.
3. In the Search filters section, select **AuthentiPad** for the Checkpoint and click **Search**.
4. In the **Search Results** table, click the **OAAM AuthenticationPad Policy** link to open the **Policy Details** page.
5. In the **Policy Details** page, click the **Rules** tab.
6. Click the **Challenge Question** link to open the Rules Details page.
7. In the Rules Details page, click the **Results** tab.
8. In the **Actions Group** list, select **OAAM HTML Pad** as the action you want triggered by this rule and click **Apply** to save the modified rule details.

   A confirmation dialog is displayed.
9. Click **OK** to dismiss the confirmation dialog.
10. Repeat Steps 5 through 9 for Challenge SMS, Registered Image and Caption, and Challenge Email rules.
11. In the Policies Detail page, click the **Trigger Combinations** tab.

12. Change the Action Group for all the Trigger Combinations to **OAAM HTML Pad** and click **Apply** after making the edits.

    A confirmation dialog is displayed.

13. Click **OK** to dismiss the confirmation dialog.

Configure OAAM Server to Use the OAAM HTML Pad

1. Follow the instructions in Section 7.2, "Adding Customizations Using the OAAM Extensions Shared Library" to extract the Extensions Shared Library.
2. Make the following changes to `oaam_custom.properties` and save it to the *`IAM_Home`*`\oaam\oaam_extensions\WEB-INF\classes\bharosa_properties` directory.

```
bharosa.uio.default.use.authentipad.checkpoint=false
bharosa.uio.default.Password.authenticator.device=DeviceHTMLControl
bharosa.uio.default.Password.authenticator.device.upgraded=DeviceHTMLControl
bharosa.uio.default.ChallengeQuestion.authenticator.device=DeviceHTMLControl
bharosa.uio.default.RegisterQuestions.authenticator.device=DeviceHTMLControl
bharosa.uio.default.ChallengeSMS.authenticator.device=DeviceHTMLControl
bharosa.uio.default.ChallengeEmail.authenticator.device=DeviceHTMLControl
```

3. Follow the instructions in Section 7.2.6, "Step 5 Repackage the OAAM Extensions Shared Library" to repackage the Extensions Shared Library, `oracle.oaam.extensions.war`.
4. Shut down all managed servers if they are running.
5. Start the WebLogic Administration Server.
6. Start the WebLogic Server where Oracle Adaptive Access Manager is deployed.
7. Follow the instructions in Section 7.2.11, "Step 10 Deploy the New OAAM Extensions Shared Library" to redeploy the OAAM Extensions Shared Library.

8.6 Questions/Answers About User Interface Customizations

A few troubleshooting tips for user interface customizations are as follows:

- **Question**: I have added the following entries to `oaam_custom.properties` in the OAAM extensions shared library:

```
bharosa.uio.default.header = /customHeader.jsp
bharosa.uio.default.footer = /customFooter.jsp
```

  OAAM server is picking up the default header and footer and not the one I specified in the extensions library.

  Answer: The custom header / footer files should have a unique name as OAAM Server pulls from the web application first. For example, `customHeader.jsp` and `customFooter.jsp`.

- **Question**: Why is the OAAM Server not picking up the `css` changes in OAAM extensions shared library?

  Answer: The property `bharosa.uio.default.custom.css` should be set to a css file that is added to the extensions library. That `css` file can override any existing CSS definitions in the base application (defined by `oaam_uio.css`).

  For example, if you want to move the username and password text and OTP pads to the center of the screen, you must set `bharosa.uio.default.custom.css =` *`CSS_`*

file_name_and_path and add the custom CSS file to the OAAM extensions shared library.

- **Question**: How do `struts_config_extension.xml` and `tiles-def-extension.xml` work in customizations?

  Answer: The OAAM extensions shared library has a `struts-config-extension.xml` and `tiles-def-extension.xml` in the `WEB-INF` folder. Any values added to these will augment or override the ones already defined by `struts-config.xml` and `tiles-def.xml` in the application.

  For example, to use a customized JSP file (`customUserPreferences.jsp`) for the base file (`userPreferences.jsp`), add the following to `tiles-def-extension.xml`:

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

- **Question**: How do I change the user name label on the OAAM Login page? The page shows:

  Sign in:

  Enter your user name.

  This is followed by the label: **Username**

  Answer: The properties which contain this data are:

```
bharosa.uio.default.username.label=UserName
bharosa.uio.default.credentials.enum.username.name=UserName
```

  Add these customizations using OAAM Extensions Shared Library. For information on customizations, see Chapter 7, "Using the OAAM Extensions Shared Library to Customize OAAM."

  Once you create extensions folder, create the file `client_resource_en.properties` inside `WEB-INF\classes` and deploy it as Shared Library to OAAM servers. Before you deploy this `oaam-extensions` war to WebLogic, make sure to delete the existing `oaam extensions`.

- **Question**: How do I change the login page completely?

  Answer: You must set `oaam.uio.login.page=`*custom_login_page* and add the file to the OAAM extensions shared library. You would need to update the `struts-config-extension.xml` contained in the OAAM extensions shared library to override the "login" outcome. Replace `oaamLoginPage.jsp` with the custom filename.

```
<action path="/entry" type="com.bharosa.uio.actions.EntryAction">
      <forward name="login" path="/oaamLoginPage.jsp" />
</action>
```

9

Customizing User Flow and Layout

The Struts/Tiles framework is used by OAAM to create a common look and feel for an application.

This chapter contains the following sections:

9.1 User Flows and Layout

The Struts configuration file `struts-config.xml` located in the `WEB-INF` directory defines all the navigation rules in the form of action definitions. The Tiles layout file `tiles-def.xml` located in the `WEB-INF` directory contains definitions for various pages.

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 (`oracle.oaam.extensions.war`). 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. Modifying the struts template is not recommended and would involve changes to both the template and the `oaamLoginPage.jsp` file.

9.1.1 Struts Actions

This section provides information about struts action definitions which are used to drive the user flow. OAAM action type classes are also summarized.

9.1.1.1 Action Definition

Action definitions typically contain path, type, and parameter attributes. The path defines what the URL will be. Many definitions also contain one or more forward elements that indicate which page should be displayed next.

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>
```

9.1.1.2 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	Updates the login status and, if appropriate, challenges the user.
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	Displays the confirmation message in OAAM Server page.
com.bharosa.uio.actions.PasswordAction	Updates the password status.
com.bharosa.uio.actions.UpdateAuthStatusAction	Updates the user authentication status and, if appropriate, it triggers pattern data processing.
com.bharosa.uio.actions.ValidateTrxAction	Validates the transaction
com.bharosa.uio.actions.FlashFingerprintAction	Fingerprints the device.
com.bharosa.uio.actions.LogoutAction	Logs out the user session and redirects to login page
com.bharosa.uio.actions.SignOnAction	Signs the user in
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	Displays Change Password link
com.bharosa.uio.actions.ForgotPasswordAction	Displays Forgot Password link
com.bharosa.uio.actions.UserInputAction	Displays input fields
com.bharosa.uio.actions.UserPreferencesDoneAction	Displays message that user completed preference registration
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	Changes the user name.
com.bharosa.uio.actions.MessageAction	Displays a message to the user
com.bharosa.uio.actions.ExitAction	Exits the user from the resource
com.bharosa.uio.actions.ErrorAction	Error occurs

9.1.2 Base Layout Definition

User interface pages are constructed using tiles in the Struts application. An external configuration file (/WEB-INF/tiles-def.xml) contains definitions for various pages.

The base layout `bharosa.uio.baseLayout` is defined to contain various sections. The header region is occupied by the `customHeader.jsp` page, the footer part is occupied by the `customFooter.jsp` page, and the body part by content. The following code shows the base layout.

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

To construct user interface pages, you define which JSP page should fill in the base layout in the `tiles-def-extension.xml` configuration file. The following example extends the `baseLayout` definition and uses a JSP named `registerQuestionsHTML.jsp` to render the content tile:

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

Tile definition can extend another Tile definition. In the `tiles-def.xml` file, you can see that only the body region changes in the user flow.

9.1.3 How Struts and Tiles Work Together

To use Tiles in the Struts application, the following extension definition was added to the `struts-config.xml` file.

```
<!-- 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>
```

Action forward entries are in the `struts-config.xml` file. When an action is forwarded to the Tile definition `baseLayout`, then the base Layout JSP page will be displayed with corresponding JSP pages in the Tile definition. For example:

```
    <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>
```

The path attribute hold the value of the Tile definition to forward. When the path value is `/registerQuestions.do` the base layout JSP page is displayed with `registerQuestionsHTML.jsp` as the body as specified in `tiles-def.xml`.

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

9.2 Custom User Flows and Layout Example

An example on how to customize the user flow and the look-and-feel of the graphical user interface is presented in the subsequent sections.

9.2.1 Customize the Look-and-Feel

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.

If the following definition is added to the `tiles-def-extension` file, the new `customPassword.jsp` is used anywhere that OAAM Server attempts to display the `password` page. The subsequent 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>
```

9.2.2 Customize the User Page Flows and Actions

To customize the user flows and actions, override the `struts` action classes and their mappings in the `struts-config-extension.xml` file.

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 use to override the login action using `struts-config-extension.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>
```

9.3 Tile Definition File

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="/customHeader.jsp"/>
    <put name="footer" value="/customFooter.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>
```

9.4 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

Configuring Properties for Localization

This chapter contains the following sections:

10.1 Turning Off Localization

There is no flag to turn-off localization, but there is a property that captures the locales supported by the deployment. You can use the property to enable one locale.

You would change the `locale.enum.XXX.adminSupported` and `locale.enum.XXX.enabled` properties to `false` for each unwanted locale.

10.2 Overriding Localized Properties

Perform customizations to localized strings in the client resource override file:

1. Create `client_resource_`*`locale`*`.properties`
2. Add property `bharosa.config.resourcebundle.clientoverride` to `client_resource_en.properties`. The default value of this property in OAAM Server is `client_resource`.

   By default the file to add custom localized strings to is `client_resource_`*`locale`*`.properties`.

   For example, for English, the file is `client_resource_en.properties`.
3. Add customized properties to `client_resource_en.properties`.
4. Create OAAM Extension WAR file containing `client_resource_en.properties` inside of `WEB-INF/classes` directory

10.3 Configuring Language Defaults for Oracle Adaptive Access Manager

You can configure language defaults in the `client_resource_`*`locale`*`.properties` file using the `bharosa.locale. enum` property. For instructions on customizing Oracle Adaptive Access Manager, see Chapter 7, "Using the OAAM Extensions Shared Library to Customize OAAM."

An example of a `bharosa.locale. enum` is shown below:

```
bharosa.locale.enum.german=2
bharosa.locale.enum.german.name=German
bharosa.locale.enum.german.description=German
bharosa.locale.enum.german.language=de
bharosa.locale.enum.german.country=
bharosa.locale.enum.german.adminSupported=true
bharosa.locale.enum.german.enabled=true
```

To enable the default locale:

1. Add and set the `bharosa.local.enum.`*`locale`*`.enabled properties` of the locales you want to support to true.
2. Add and set the `bharosa.local.enum.`*`locale`*`.enabled` properties of the locales you do not want to support to false.
3. Add and set the `bharosa.default.locale property` to match the `bharosa.locale.enum.`*`locale`* property of your locale.

> **Note:** The only locales supported are the ones listed in the enums.

10.3.1 Example 1

A German bank wants to set German as the default language and wants to support only German. Follow these steps to configure the `client_resource_de.properties`:

1. If the locale enum does not exist, create it:

   ```
   bharosa.locale.enum.german.enabled=true
   ```

2. If the locale enum already exists, set it to true.
3. If present, set other `bharosa.local.enum.`*`locale`*`.enabled` properties to `false`.

   ```
   bharosa.locale.enum.italian.enabled=false
   bharosa.locale.enum.french.enabled=false
   bharosa.locale.enum.portuguese_br.enabled=false
   bharosa.locale.enum.spanish.enabled=false
   bharosa.locale.enum.korean.enabled=false
   bharosa.locale.enum.chinese_cn.enabled=false
   bharosa.locale.enum.chinese_tw.enabled=false
   bharosa.locale.enum.japanese.enabled=false
   bharosa.locale.enum.arabic.enabled=false
   bharosa.locale.enum.czech.enabled=false
   bharosa.locale.enum.danish.enabled=false
   bharosa.locale.enum.dutch.enabled=false
   bharosa.locale.enum.finnish.enabled=false
   bharosa.locale.enum.greek.enabled=false
   bharosa.locale.enum.hebrew.enabled=false
   bharosa.locale.enum.hungarian.enabled=false
   bharosa.locale.enum.norwegian.enabled=false
   bharosa.locale.enum.polish.enabled=false
   bharosa.locale.enum.portuguese.enabled=false
   bharosa.locale.enum.romanian.enabled=false
   bharosa.locale.enum.russian.enabled=false
   bharosa.locale.enum.slovak.enabled=false
   bharosa.locale.enum.swedish.enabled=false
   bharosa.locale.enum.thai.enabled=false
   bharosa.locale.enum.turkish.enabled=false
   ```

4. Set `bharosa.default.locale property` to match the value of the locale enum.

   Since `bharosa.locale.enum.german=2`, set `bharosa.default.locale` property to 2.

   If the property does not exist, create it.

10.3.2 Example 2

A Brazilian bank wants to set Brazilian Portuguese as the default, but wants to display all the other languages that OAAM Server had been translated to. To configure the setting:

1. If the locale enum does not exist, create it:

   ```
   bharosa.locale.enum.pt_br.enabled=true
   ```

2. If the locale enum already exists, set it to true.
3. Set all other `bharosa.local.enum.`*`locale`*`.enabled` properties using the Properties Editor to false.
4. Set `bharosa.default.locale` property to the value of the locale enum using the Properties Editor.

   If `bharosa.locale.enum.pt_br=9`, set `bharosa.default.locale property` to `9`.

5. Set `bharosa.locale.enum.`*`locale`*`.enabled` property in `client_resource_`*`locale`*`.properties` for all the languages OAAM Server had been translated to and ensure they are set to true.

   ```
   bharosa.locale.enum.german.enabled=true
   bharosa.locale.enum.italian.enabled=true
   bharosa.locale.enum.french.enabled=true
   bharosa.locale.enum.portuguese_br.enabled=true
   bharosa.locale.enum.spanish.enabled=true
   bharosa.locale.enum.korean.enabled=true
   bharosa.locale.enum.chinese_cn.enabled=true
   bharosa.locale.enum.chinese_tw.enabled=true
   bharosa.locale.enum.japanese.enabled=true
   bharosa.locale.enum.arabic.enabled=true
   bharosa.locale.enum.czech.enabled=true
   bharosa.locale.enum.danish.enabled=true
   bharosa.locale.enum.dutch.enabled=true
   bharosa.locale.enum.finnish.enabled=true
   bharosa.locale.enum.greek.enabled=true
   bharosa.locale.enum.hebrew.enabled=true
   bharosa.locale.enum.hungarian.enabled=true
   bharosa.locale.enum.norwegian.enabled=true
   bharosa.locale.enum.polish.enabled=true
   bharosa.locale.enum.portuguese.enabled=true
   bharosa.locale.enum.romanian.enabled=true
   bharosa.locale.enum.russian.enabled=true
   bharosa.locale.enum.slovak.enabled=true
   bharosa.locale.enum.swedish.enabled=true
   bharosa.locale.enum.thai.enabled=true
   bharosa.locale.enum.turkish.enabled=true
   ```

6. Set `bharosa.default.locale property` in `client_resource_`*`locale`*`.properties` to `9`.

10.3.3 Example 3

A French bank wants clients to see French as a default, and wants to support only French, German, English, and Italian. The French locale enum is already present in the `client_resource_fr.properties` file.

```
bharosa.locale.enum.french=5
bharosa.locale.enum.french.name=French
bharosa.locale.enum.french.description=French
bharosa.locale.enum.french.language=fr
bharosa.locale.enum.french.country=
bharosa.locale.enum.french.adminSupported=true
bharosa.locale.enum.french.enabled=true
```

To configure the application:

1. In `client_resource_fr.properties` set `bharosa.locale.enum.`*`locale`*`.enabled` to `true` for German, Italian, and English.

   ```
   bharosa.locale.enum.german.enabled=true
   bharosa.locale.enum.italian.enabled=true
   bharosa.locale.enum.english.enabled=true
   ```

2. Set all other `bharosa.local.enum.`*`locale`*`.enabled` properties to `false`.
3. Set `bharosa.default.locale` property to the value of the locale enum.

   Since `bharosa.locale.enum.french=5`, set `bharosa.default.locale` property to `5`.

10.3.4 Example 4

A German bank wants to set English as the default language and wants to support all other languages. To do this, follow these steps for `client_resource_de.properties`:

1. If the locale enum does not exist, create it:

   ```
   bharosa.locale.enum.english.enabled=true
   ```

2. If the locale enum already exists, set it to `true`.
3. If present, set other `bharosa.local.enum.`*`locale`*`.enabled` properties to true.

   ```
   bharosa.locale.enum.italian.enabled=true
   bharosa.locale.enum.german.enabled=true
   bharosa.locale.enum.french.enabled=true
   bharosa.locale.enum.portuguese_br.enabled=true
   bharosa.locale.enum.spanish.enabled=true
   bharosa.locale.enum.korean.enabled=true
   bharosa.locale.enum.chinese_cn.enabled=true
   bharosa.locale.enum.chinese_tw.enabled=true
   bharosa.locale.enum.japanese.enabled=true
   bharosa.locale.enum.arabic.enabled=true
   bharosa.locale.enum.czech.enabled=true
   bharosa.locale.enum.danish.enabled=true
   bharosa.locale.enum.dutch.enabled=true
   bharosa.locale.enum.finnish.enabled=true
   bharosa.locale.enum.greek.enabled=true
   bharosa.locale.enum.hebrew.enabled=true
   bharosa.locale.enum.hungarian.enabled=true
   ```

```
bharosa.locale.enum.norwegian.enabled=true
bharosa.locale.enum.polish.enabled=true
bharosa.locale.enum.portuguese.enabled=true
bharosa.locale.enum.romanian.enabled=true
bharosa.locale.enum.russian.enabled=true
bharosa.locale.enum.slovak.enabled=true
bharosa.locale.enum.swedish.enabled=true
bharosa.locale.enum.thai.enabled=true
bharosa.locale.enum.turkish.enabled=true
```

4. Set `bharosa.default.locale property` to match the value of the locale enum.

   Since `bharosa.locale.enum.english=0`, set `bharosa.default.locale` property to 0.

   If the property does not exist, create it.

10.4 Customizing Abbreviations and Equivalences for Locales

Oracle Adaptive Access Manager supports the concept of "fuzzy logic." Fuzzy logic, in part, relies on preconfigured sets of word equivalents, commonly known as abbreviations.

In the English version of Oracle Adaptive Access Manager, there are several thousand English abbreviations (and equivalences).

In all other languages, it is necessary for the installer to enhance the brief abbreviation files provided. Without additions, the fuzzy logic will be not as effective.

Locale-specific abbreviation files are shipped with OAAM. These files are named `bharosa_auth_abbreviation_config_`*`locale`*`.properties` where *`locale`* is the locale string. For example, the Spanish version of the file is `bharosa_auth_abbreviation_config_es.properties`.

Changes cannot be made to this file. To customize abbreviations, a new file must be created with a new set of abbreviations. This file takes precedence over the original file and all abbreviations in the original file are ignored.

To localize for one locale (for example, for Japanese only), perform the following steps:

1. Create one file specific to the locale with the same prefix as the original locale-specific abbreviation file. For example, `Abbreviations_ja.properties` for Japanese.
2. Add the file to the OAAM Extensions Shared Library (`WEB-INF/classes`).
3. Using OAAM, set the value of property `bharosa.authenticator.AbbreviationFileName` to that file's absolute path, `WEB-INF/classes/Abbreviations_ja.properties` in the extensions folder.

If you want customize for multiple locales, perform the following steps:

1. Create the files specific to those locales with the same prefix as the original locale-specific abbreviation file.

   For example,

   `/mydrive/IDM_ORACLE_HOME/oaam/conf/Abbreviations_es.properties` for Spanish

   `/mydrive/IDM_ORACLE_HOME/oaam/conf/Abbreviations_ja.properties` for Japanese
2. Add the file to the OAAM Extensions Shared Library (`WEB-INF/classes`).

3. Using OAAM, set the value of property `bharosa.authenticator.AbbreviationFileName` to that file's absolute path, `WEB-INF/classes/Abbreviations.properties` in the extensions folder.

   The locale prefix is absent in the value of the property because the locale settings of the end user's browser determine the run-time locale.

11

Setting Up Custom 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.

This chapter describes the initial steps you must perform to set up custom device fingerprinting.

This chapter contains the following sections:

11.1 Out of the Box Fingerprint Types

There are two out of box fingerprint types available:

- Flash
- Applet

For most typical deployments, default OAAM fingerprinting satisfies client requirements, but you may want to set OAAM to perform custom fingerprinting. For information on setting up custom fingerprinting, see the section following.

11.2 Setting Up Custom Fingerprinting

This chapter provides information on how to create fingerprint types so that Oracle Adaptive Access Manager can capture information about the devices that a user utilizes when accessing protected applications. Fingerprint types are contained in the `oaam_custom.properties`. If you want fingerprint types that are not provided out of the box, you must modify your `oaam_custom.properties` file to include these types at the time of deployment.

1. Open the `oaam_custom.properties` file in the `WEB-INF/classes/bharosa_properties` directory of the `oracle.oaam.extensions.war` file.
2. Add the enumeration for the fingerprint you want to capture.

   Examples of the fingerprint type enum are as follows:

```
vcrypt.fingerprint.type.enum=Enum for fingerprint type
vcrypt.fingerprint.type.enum.browser=1
vcrypt.fingerprint.type.enum.browser.name=Browser
vcrypt.fingerprint.type.enum.browser.description=Browser
```

```
vcrypt.fingerprint.type.enum.browser.userAgent=userAgent
vcrypt.fingerprint.type.enum.browser.locallang=localLang
vcrypt.fingerprint.type.enum.browser.localcountry=localCountry
vcrypt.fingerprint.type.enum.browser.localvariant=localVariant
vcrypt.fingerprint.type.enum.browser.header_list=
       locallang,localcountry,localvariant,userAgent
vcrypt.fingerprint.type.enum.browser.search_list=locallang,userAgent
vcrypt.fingerprint.type.enum.browser.result_list=locallang,userAgent
vcrypt.fingerprint.type.enum.browser.header_value_nv=t,true,f,false,en,English,
  es,Spanish,de,German,it,Italian,ja,Japanese,fr,French,ko,Korean,
  zh,Chinese,ar,Arabic,cs,Czech,da,Danish,nl,Dutch,fi,Finnish,el,Greek,
  iw,Hebrew,hu,Hungarian,no,Norwegian,pl,Polish,pt,Portuguese,ro,Romanian,
  ru,Russian,sk,Slovak,sv,Swedish,th,Thai,tr,Turkish,BR,Brazil

vcrypt.fingerprint.type.enum.flash=2
vcrypt.fingerprint.type.enum.flash.name=Flash
vcrypt.fingerprint.type.enum.flash.description=Flash
vcrypt.fingerprint.type.enum.flash.processor=
  com.bharosa.uio.processor.device.FlashDeviceIdentificationProcessor
vcrypt.fingerprint.type.enum.flash.header_list=
avd,acc,a,ae,ev,ime,mp3,pr,sb,sp,sa,sv,tls,ve,deb,l,lfd,m,os,ar,pt,col,dp,r,v
vcrypt.fingerprint.type.enum.flash.search_list=deb,l,os,v
vcrypt.fingerprint.type.enum.flash.result_list=deb,l,os,v
vcrypt.fingerprint.type.enum.flash.header_name_nv=
   avd,Audio/Video disabled by user,
   acc,Has accessibility,a,Has audio,ae,Had audio encoder,ev,Embedded video,
   ime, Has input method editor (IME) installed,mp3, Has MP3,
   pr, Supports printer, sb, Supports screen broadcast applications,
   sp, Supports playback on screen broadcast applications,
   sa, Supports streaming audio, sv, Supports streaming video,
   tls, Supports native SSL, ve, Contains video encoder,
   deb, Debug version, l, Language, lfd, Is local file read disabled,
   m, Manufacturer, os, Operating System, ar, Aspect ratio of screen,
   pt, Player type, col, Is screen color,
   dp, Dots-per-inch (DPI), r, Screen resolution, v, Flash version

#vcrypt.fingerprint.type.enum.flash.header_value_nv=t,true,f,false
vcrypt.fingerprint.type.enum.flash.header_value_nv=
   t,true,f,false,en,English,es,Spanish,de,German,it,Italian,
   ja,Japanese,fr,French,ko,Korean,zh,Chinese,ar,Arabic,
   cs,Czech,da,Danish,nl,Dutch,fi,Finnish,el,Greek,
   iw,Hebrew,hu,Hungarian,no,Norwegian,pl,Polish,pt,Portuguese,ro,Romanian,
   ru,Russian,sk,Slovak,sv,Swedish,th,Thai,tr,Turkish,BR,Brazil

vcrypt.fingerprint.type.enum.flash.avd=Audio/Video disabled by user
vcrypt.fingerprint.type.enum.flash.acc=Has accessibility
vcrypt.fingerprint.type.enum.flash.a=Has audio
vcrypt.fingerprint.type.enum.flash.ae=Had audio encoder
vcrypt.fingerprint.type.enum.flash.ev=Embedded video
vcrypt.fingerprint.type.enum.flash.ime= Has input method editor (IME) installed
vcrypt.fingerprint.type.enum.flash.mp3= Has MP3
vcrypt.fingerprint.type.enum.flash.pr= Supports printer
vcrypt.fingerprint.type.enum.flash.sb= Supports screen broadcast applications
vcrypt.fingerprint.type.enum.flash.sp=
    Supports playback on screen broadcast applications
vcrypt.fingerprint.type.enum.flash.sa= Supports streaming audio
vcrypt.fingerprint.type.enum.flash.sv= Supports streaming video
vcrypt.fingerprint.type.enum.flash.tls= Supports native SSL
vcrypt.fingerprint.type.enum.flash.ve= Contains video encoder
vcrypt.fingerprint.type.enum.flash.deb= Debug version
```

```
vcrypt.fingerprint.type.enum.flash.l= Language
vcrypt.fingerprint.type.enum.flash.lfd= Is local file read disabled
vcrypt.fingerprint.type.enum.flash.m= Manufacturer
vcrypt.fingerprint.type.enum.flash.os= Operating System
vcrypt.fingerprint.type.enum.flash.ar= Aspect ratio of screen
vcrypt.fingerprint.type.enum.flash.pt= Player type
vcrypt.fingerprint.type.enum.flash.col= Is screen color
vcrypt.fingerprint.type.enum.flash.dp= Dots-per-inch (DPI)
vcrypt.fingerprint.type.enum.flash.r= Screen resolution
vcrypt.fingerprint.type.enum.flash.v= Flash version

vcrypt.fingerprint.type.enum.monitordata=3
vcrypt.fingerprint.type.enum.monitordata.name=MonitorData
vcrypt.fingerprint.type.enum.monitordata.description=Monitor Data

vcrypt.fingerprint.type.enum.applet=999
vcrypt.fingerprint.type.enum.applet.name=Applet
vcrypt.fingerprint.type.enum.applet.description=Applet
vcrypt.fingerprint.type.enum.applet.processor=
    com.bharosa.uio.processor.device.AppletDeviceIdentificationProcessor
vcrypt.fingerprint.type.enum.applet.header_list=
    java.version,java.vendor,os.name,os.arch,os.version
vcrypt.fingerprint.type.enum.applet.header_name_nv=
    java.version,Java Version,
    java.vendor,Java Vendor Name,os.name,Operating System Name,
    os.arch,Operating System Architecture,
    os.version,Operating System Version

vcrypt.fingerprint.type.enum.applet.header_value_nv=t,true,f,false
vcrypt.fingerprint.type.enum.native_mobile=900
vcrypt.fingerprint.type.enum.native_mobile.name=Native Mobile
vcrypt.fingerprint.type.enum.native_mobile.description=
    Native Mobile implementation using OIC
vcrypt.fingerprint.type.enum.native_mobile.processor=
  com.bharosa.uio.processor.device.NativeMobileDeviceIdentificationProcessor
vcrypt.fingerprint.type.enum.native_mobile.header_list=
  os.type,os.version,hw.imei,hw.mac_addr
vcrypt.fingerprint.type.enum.native_mobile.header_name_nv=
  os.type,Operating System Type,os.version,Operating System Version,
  hw.imei,Hardware IMEI Number,hw.mac_addr,Hardware Mac Address
vcrypt.fingerprint.type.enum.native_mobile.header_value_nv=t,true,f,false
```

3. Set the property `bharosa.uio.default.device.identification.scheme` to the type of fingerprint you want to capture.

   For example, the `vcrypt.fingerprint.type.enum` elementId for digital device fingerprinting is:

```
bharosa.uio.default.device.identification.scheme=flash
```

12

Flash Fingerprinting in Native Integration

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.

This chapter contains the following sections:

12.1 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.

12.2 Definitions of Variables and Parameters

Table 12–1 lists the parameter and response variable in the interaction between the Flash movie and the application.

Table 12–1 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 machine accessible to the Flash player.

12.3 Implementations of Flash Fingerprinting

This section contains information about the various implementations of Flash fingerprinting.

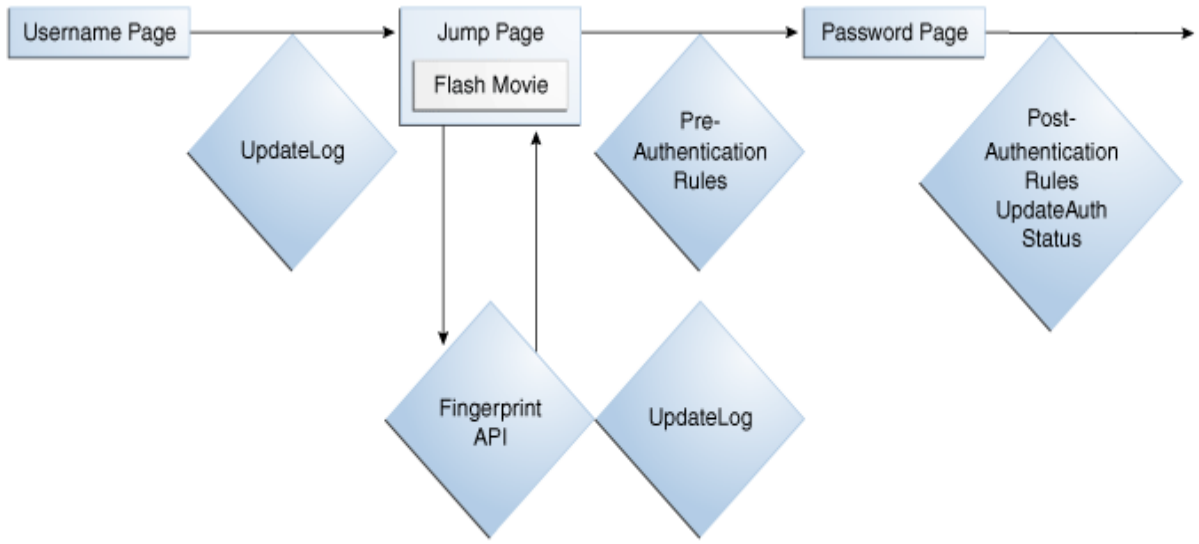
12.3.1 Option 1

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.

12.3.1.1 Option 1 Flow

Figure 12–1 shows the flow of Option 1.

Figure 12–1 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

12.3.1.2 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
```

12.3.2 Option 2

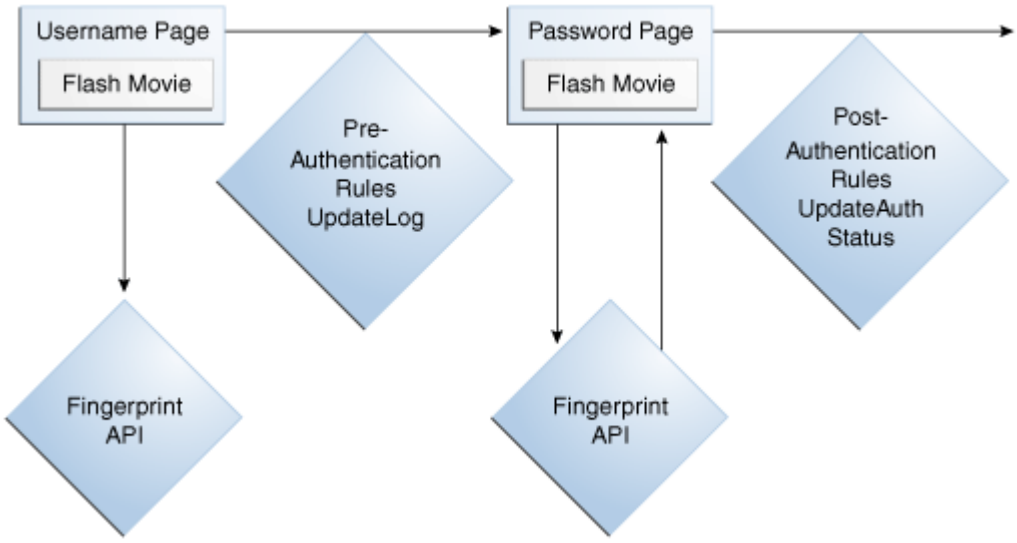
Option 2 is a newer, more streamlined user experience that eliminates the Jump Page from the user experience. To do this, the Flash movie is included in both the user name page and the password page.

The first movie (username page) is used to accept the existing value of the Flash cookie and fingerprint, but no new value is set. After the user is loaded and the OAAM session is created a second movie is presented (password page) where the new values for the session are then set back to the client machine.

12.3.2.1 Option 2 Flow

Figure 12–2 shows the flow of Option 2.

Figure 12–2 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

12.3.2.2 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 occurred, using previously
                       // generated cookie set
                    }
                 return cookieSet.getFlashCookie();
             } catch (Exception e) {
                   // Handle fingerprinting error
             }
             return "";
} // flashFingerPrint
```

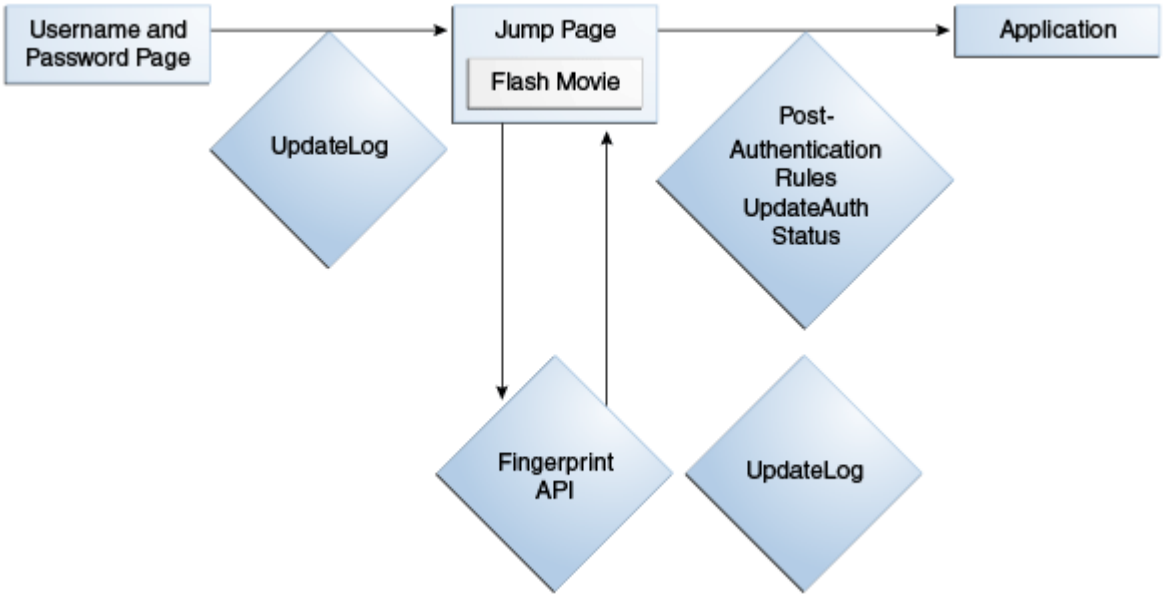
12.3.3 Option 3

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.

12.3.3.1 Option 3 Flow

Figure 12–3 shows the flow of Option 3.

Figure 12–3 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

12.3.3.2 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
```

12.3.3.3 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;
}
```

12.4 Flash Fingerprinting Included in Web Application with Native Integration

Instructions to implement Flash fingerprinting is as follows:

The native integration OAAM sample application uses a parameter called `dcPurp` to post the Flash movie to `handleFlash.jsp`.

There are three possible values for `dcPurp`:

- sample - this will post to `handleFlash.jsp`
- native - this will post to `dc`, where `dc` is expected to be configured as a java servlet to accept the fingerprint post. You can use the OAAM class `CookieManager.java` for this purpose
- uio - this will post to `flashFingerprint.do`
- .net - this will post to `CookieManager.aspx`

If no value is passed for `dcPurp`, then `dc` is used.

The subsequent sample code is provided for your reference.

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
          width="1" height="1" id="flash" align="middle">
    <param name="allowScriptAccess" value="sameDomain"/>
    <param name="movie" value="<%= redirect_flashSrc %>"/>
    <param name="quality" value="low"/>
    <param name="bgcolor" value="#ffffff"/>
    <param name="FlashVars" value="dcPurp=sample"/>
    <embed src="<%= redirect_flashSrc %>" quality="low"
```

```
          bgcolor="#ffffff" FlashVars="dcPurp=sample" width="1" height="1"
          name="flash" align="middle"
          allowScriptAccess= "sameDomain"
          type="application/x-shockwave-flash"/>
</object>
```

The `FlashVars` key / value pair of: `dcPurp=sample` indicates that the Flash movie should post to `handleFlash.jsp`.

13

Extending Device Identification

This chapter describes how to extend device identification in a typical deployment. It includes the following sections:

13.1 What is Device Identification?

A device is a computer, PDA, cell phone, kiosk, and so on, a user uses to login from a location. A Device ID is an internal Oracle Adaptive Access Manager identifier for the device used for the login session. This section provides information on the process of identifying the device and assigning a Device ID to it.

The process involves three stages:

- Data Gathering
- Data Processing
- Data Storage

13.1.1 Data Gathering

Device identification is a mechanism to recognize the devices a customer uses. The fingerprint details can help in identifying a device, check whether it is secure, and determine the risk level for the authentication or transaction. During the login process, data is collected such as browser type, browser headers, operating system type, locale, and so on from the device fingerprint.

There are two categories of data: secure and digital. Each of these categories have within them a fingerprint and a cookie. Oracle Adaptive Access Manager uses two types of cookies to perform device identification. One is the secure cookie (also known as browser cookie) and the other is the digital cookie (also known as the flash cookie).

- Secure data is gathered from the user's browser. This data includes the user-agent string, and an HTTP cookie value. The User-Agent is used as the secure fingerprint. The HTTP cookie value is a unique one-time use cookie that is set every time a user logs in. This cookie value is retrieved from the user's browser upon login.
- Digital data is gathered from the user's Adobe Flash installation. This data includes an array of Flash system capability data, and a Flash Locally Stored Object (LSO). The Flash capability data is used as the digital fingerprint representing the Flash system capabilities. The LSO contains a unique one-time

use value that is set every time a user logs in. This value is retrieved using a flash object that runs upon login.

An example of data gathering flow is:

1. User enters user name.
2. Login jump page is displayed. This second login page is a transient page to capture the flash and secure cookies and for fingerprinting the user device.
3. Pre-authentication checkpoint. Pre-authentication rules are run before the user is authenticated. Common values returned by the pre-authentication checkpoint include: Allow and Block.
4. AuthentiPad checkpoint. This stage determines the virtual authentication device to use.
5. Password page is displayed.

13.1.2 Data Processing

Once this data is gathered, the OAAM Server must process the device fingerprint data and determine if this device has ever been seen before. Oracle Adaptive Access Manager does not solely rely on one element to develop the "device fingerprint". If persistent cookies are disabled, Oracle Adaptive Access Manager still has other information to use in identifying the device. For example, in the case where datapoints are unavailable, the system can still provide robust security utilizing other objects (secure cookie, flash cookie, HTTP header, Real Media, QuickTime, and so on). Based on the presence of the datapoints it determines if this is a new, or existing device.

Device fingerprinting generates a fingerprint for each user session which is unique to the individual's device and which is replaced upon each subsequent visit with another unique fingerprint.

The Device ID is generated when a device matching the data available in the fingerprint cannot be found in the OAAM database. The process of looking for existing devices based on device fingerprint data includes searching for devices with matching rotating cookie values and fingerprints. OAAM weights fingerprint data differently, and the weighting is roughly as follows: Flash Cookie, Flash Fingerprint, Secure Cookie, Secure Fingerprint. If a device is found that matches the criteria, no new device id will be generated.

This process can be augmented, or modified by creating policies in the Device Identification checkpoint. This checkpoint is run during device fingerprinting to allow for additional rules to be processed to influence device identification also assess risk. The risk score that is calculated during the device identification checkpoint evaluation is used as the Device Risk Gradient in the pre-conditions of rules configured in other checkpoint's policies.

The device risk gradient specifies the certainty of the device being identified. It is standard in most rules as a pre-condition. For example, a device risk gradient of 0 is an exact match whereas a device gradient of 500 is a "similar" device, and a score of 1000 a "different" device.

13.1.3 Data Storage

Once a device has been given an ID, new rotating cookie values are generated and set. The secure cookie is set as an HTTP cookie, and the digital cookie is set as a Flash LSO by the flash object. These two values are the only values stored on a user's computer during the device identification process.

13.2 When to Extend Device Identification

For most typical deployments, the out-of-the-box device identification satisfies client requirements, but you may be looking to have the ability to extend that process and include additional information in the fingerprint. 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. For example, iPhone, iPad, and so on.
- There is a need to extract stronger device identification data from the client using a non-flash extension that can run inside the browser

Starting from the 11.1.1.5 release of OAAM a framework exists that you can use to extend device identification and implement in both native integrations, and non-native integrations. The framework is separated into the client side extension, and the OAAM server device identification extension.

13.2.1 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 extension 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. For information on using the OAAM Extensions Shared Library, see Chapter 7, "Using the OAAM Extensions Shared Library to Customize OAAM."

13.2.2 Developing a Custom Device Identification Extension

The custom device identification extension is software that extends the out-of-the-box device identification provided by Oracle Adaptive Access Manager.

13.2.2.1 Implement the Client Side Extension

Implementing the client side extension that can run in the client browser involves coding the extension using the appropriate technology.

The client side extension can be implemented in any technology if it can 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.**

  The technology chosen to implement the client side extension must run in the context of the user's browser. Technologies such as Flash, JavaScript, Java Applets are typical choices.

- **It can communicate with OAAM Server and post data using the HTTP protocol**.

  The fingerprint data and rotating cookie must be sent to the OAAM server if the standard integration is using the HTTP POST method.

- **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 data sent to the server must be sent in the context of the user's session in order for the fingerprinting data to be associated with the user's login. The presence of a valid JSESSIONID is required for this to work.

- **The list of data/values that are collected by the extension uniquely identifies a client device**.

  In general the fingerprints collected by OAAM should be as unique as possible given the data constraints imposed by the user's device. When extending device identification, this is the best opportunity to gather additional data to uniquely identify a user's device.

- **The extension can retrieve and store a cookie equivalent on the client computer**.

  The concept of a rotating one time use cookie is core to the device fingerprinting process. The device identification must support the capability to store and retrieve the value provided to the extension by the OAAM server.

- **The extension can submit the following parameters to flashFingerprint.do URL on OAAM Server using HTTP Post**:

  Note: This requirement is only relevant when using the standard OAAM implementation.

Table 13–1 Parameters to flashFingerprint.do URL

Name of the parameter	Description
client	Name of the client extension. A constant value that indicates the extension 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 extension 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 extension.

13.2.2.2 Add Properties Related to Custom Device Identification Extension to OAAM Extensions Shared Library

The static values are related to the properties that need to be defined within the OAAM Server to make it aware of the new extension.

To create custom fingerprint types, proceed as follows:

1. Open the `oaam_custom.properties` file of the OAAM Extensions Shared Library war.
2. Add the following properties as enum element to `vcrypt.fingerprint.type.enum`.

   Note: Replace *`extension-name`* with a string that represents your extension. Do not use the strings 'flash', 'browser' as they are already used by the OAAM product.

Table 13–2 vcrypt.fingerprint.type.enum elements

Property Name	Value Description
vcrypt.fingerprint.type.enum.*extension-name*	Integer value above 100
vcrypt.fingerprint.type.enum. *extension-name*.name	Name that represents the extension
vcrypt.fingerprint.type.enum. *extension-name*.description	Description of the extension
vcrypt.fingerprint.type.enum. *extension-name*.processor	Fully qualified java class name of the processor class that implements device identification logic on the server side. See the next section for details on how to implement this class.
vcrypt.fingerprint.type.enum. *extension-name*.header_list	Comma separated list of data that is collected by the client side extension.
vcrypt.fingerprint.type.enum. *extension-name*.header_name_nv	Comma separated list of data and readable name of those data.
vcrypt.fingerprint.type.enum. *extension-name*. header_value_nv	Comma separated list of mappings of value to readable string of those values
bharosa.uio.default.device.identification.scheme	*extension-name* Note: This is very important for OAAM to use the custom device identification

3. Set the fingerprint scheme to the fingerprint type enum element ID/key.

   For example:

   bharosa.uio.default.device.identification.scheme=flash

Example

The following flash fingerprint type is shown as an example.

```
vcrypt.fingerprint.type.enum.flash=2
vcrypt.fingerprint.type.enum.flash.name=Flash
vcrypt.fingerprint.type.enum.flash.description=Flash
vcrypt.fingerprint.type.enum.flash.processor=
    com.bharosa.uio.processor.device.FlashDeviceIdentificationProcessor
vcrypt.fingerprint.type.enum.flash.header_list=
    avd,acc,a,ae,ev,ime,mp3,pr,sb,sp,sa,sv,tls,ve,deb,l,lfd,m,os,ar,pt,col,dp,r,v
vcrypt.fingerprint.type.enum.flash.search_list=deb,l,os,v
vcrypt.fingerprint.type.enum.flash.result_list=deb,l,os,v
vcrypt.fingerprint.type.enum.flash.header_name_nv=
    avd,Audio/Video disabled by user,acc,Has accessibility,
    a,Has audio,ae,Had audio encoder,ev,Embedded video,
    ime, Has input method editor (IME) installed,
    mp3, Has MP3, pr, Supports printer,
    sb, Supports screen broadcast applications,
    sp, Supports playback on screen broadcast applications,
    sa, Supports streaming audio,
    sv, Supports streaming video, tls, Supports native SSL,
    ve, Contains video encoder, deb, Debug version, l, Language, lfd,
    Is local file read disabled, m, Manufacturer, os, Operating System,
    ar, Aspect ratio of screen, pt, Player type, col, Is screen color,
    dp, Dots-per-inch (DPI), r, Screen resolution, v, Flash version
#vcrypt.fingerprint.type.enum.flash.header_value_nv=t,true,f,false

vcrypt.fingerprint.type.enum.flash.header_value_nv=
    t,true,f,false,en,English,es,Spanish,de,German,it,Italian,ja,Japanese,
```

```
fr,French,ko,Korean,zh,Chinese,ar,Arabic,cs,Czech,
da,Danish,nl,Dutch,fi,Finnish,el,Greek,iw,Hebrew,hu,Hungarian,
no,Norwegian,pl,Polish,pt,Portuguese,ro,Romanian,ru,Russian,
sk,Slovak,sv,Swedish,th,Thai,tr,Turkish,BR,Brazil
```

13.2.2.3 Extend/Implement the DeviceIdentification Extension Class

Extend the `DeviceIdentification` extension class: `com.bharosa.uio.processor.device.DeviceIdentificationProcessorBase` and implement the methods documented in this section. The server-side extension extends all of the required methods.

13.2.2.3.1 getPlugInHTML `public String getPlugInHTML();`

Implementation should return a valid extension 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 of extension HTML is shown as follows:

```
<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.

13.2.2.3.2 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 extension.

This method is called when the client side extension 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 extension.

As mentioned in the previous section, client extension 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.

13.2.2.3.3 getDigitalCookie

```
public String getDigitalCookie();
```

Implementation should return the digital cookie sent by the client extension. 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 extension.

13.2.2.3.4 getClientDataMap

```
public Map getClientDataMap(HttpServletRequest request);
```

Implementation should read the data from the request and store it into a map that you can use for logging or auditing purposes.

13.2.3 Overview of Interactions

Following is the overview of how the device identification extension 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 extension class. It then asks the extension class for the HTML that must be embedded in the user login page. The OAAM Server returns the user login page with the device identification extension HTML.
3. Once the login page is rendered, the client based extension is activated and collects information about the device.
4. The client extension then submits the collected data to the device identification URL on the OAAM Server.
5. The OAAM Server then calls the device identification extension to obtain the fingerprint based on collected data from the client extension.
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 extension 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 extension so that it is stored on the client system.
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, you can use the digital cookie to look up the device without having to fingerprint.

13.2.4 Compile, Assemble and Deploy

Compile the custom device identification extension class and assemble the OAAM Extensions Shared library. For instructions on deploying the OAAM extensions shared library, see Chapter 7, "Using the OAAM Extensions Shared Library to Customize OAAM."

13.2.5 Important Note About Implementing the Extension

When implementing the extension, keep the following points in mind:

- Make sure the custom device identification class outputs a valid HTML required to activate the client side extension.
- Make sure the client side extension posts the data to OAAM Server using the "existing HTTP Session".

14

Enabling 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.

This chapter contains the following sections:

14.1 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

14.2 Enabling Device Registration Out-of-the-Box

In Oracle Adaptive Access Manager out-of-the-box, to enable device registration so users can register their devices for all applications:

To enable the device registration option for users, add the following properties to `oaam_custom.properties`:

Properties	Description
bharosa.uio.default.register.questions.registerdevice.enabled	Adds device registration to the challenge question registration page. Set to true.

Properties	Description
bharosa.uio.default.register.userinfo.registerdevice.enabled	Adds device registration to the Contact Information registration page Set to true.
bharosa.uio.default.registerdevice.enabled	Enables device registration Set to true

Note: To enable the features on an application-specific bases, `default` can be replaced with the appropriate `appId` in each of the prior property names.

To enable the unregistering of device(s) option from User Preferences, add the following properties:

Properties	Description
bharosa.uio.default.userpreferences.unregister.this.enabled	Enables user to be able to unregister current device in user preferences Set to true
bharosa.uio.default.userpreferences.unregister.all.enabled=true	Enables user to be able to unregister all devices in user preferences Set to true

If you set these properties, the user can choose to register his device in the user preferences on the OAAM server.

14.3 Create Policies to Use Device Information

Once the feature is enabled, information about the device is collected for that user. 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.

14.4 CSR Resetting Device 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. These actions are also available in the user preferences in OAAM server.

Part IV

Integrating Applications

Part IV contains the following chapters:

15

Integrating Client Applications with OAAM for Transactions

Oracle Adaptive Access Manager can evaluate the risk associated with a transaction in real-time to prevent fraud and misuse. Any user activity that requires monitoring after successfully logging in can be termed as a transaction.

This chapter covers the integration of native client applications with OAAM for the risk analysis of transactions. It includes the following sections:

15.1 Transaction Example

An example of a transaction is an e-commerce transaction in which the buyer purchases a book over the Internet. John Doe logs on to his laptop, accesses the "Bigbookemporium.com" Internet site and performs an electronic search. John selects the book he wants and Bigbookemporium.com displays the purchase price. When John is ready to check out, he pays for it by credit card by providing the following information:

- Amount of the transaction
- Credit card type
- Credit card number
- Credit card expiration date
- Buyer first and last name
- Buyer billing address

15.2 About the Transaction Flow

Figure 15–1 shows the transaction flow when the OAAM Server processes client transactions for risk. Numbers correspond to numbers in the figure.

Figure 15–1 Tradition Transaction Flow With OAAM Server Added

1. At your website, the customer submits a transaction.
2. You invoke the `Create Transaction` OAAM API along with transaction data to create a transaction in OAAM database.

   Note: The `CreateTransaction` API requires the Session ID to create a transaction. If you do not have a Session ID, you must call the `CreateOAAMSession` API to create a session before calling the `CreateTransaction` API.

3. The OAAM API returns the status of transaction creation to you.
4. Based on the business requirements, you invoke a `Run Rules` API to analyze the risk/fraud related to the customer transaction.
5. OAAM server returns the rules response that contains action(s) and score to you.
6. Based on the OAAM response, you decide on a corresponding action (Allow, Block, Challenge, and so on) and the website displays the result to the customer.

15.3 High-Level Steps Required to Integrate Native Client Applications with OAAM

The high-level steps required to integrate the native client applications with OAAM are as follows:

1. Identify the types of client transactions that need to be integrated with OAAM. Prepare a list of transaction specific attributes (data) that makes up the transaction. Refer to this as "Source" data or Client Data. For example, in an online transaction, the data involved may be credit cards, e-checks, debit cards, dollar amounts, name, shipping and billing addresses.
2. Design and develop the transaction and entity definitions in OAAM. The entities and transaction data elements are then mapped to the source data (client-specific data) so that the OAAM Server can process the information from the client application. For additional information on modeling a transaction in OAAM, see *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*.

3. Identify and determine the checkpoints required based on the types of transactions.
4. Define and configure the security policies and rules for risk evaluation that are specific to each of the transactions.
5. Write code in the client application to call OAAM APIs to create or update the transactions, evaluate risk, and also to receive specific responses from OAAM.
6. Test the integration and make sure it is working end-to-end.
7. For information on how the integration and risk evaluation work for different transactions and the usage of the transaction APIs, see the out-of-the-box OAAM Sample application.

   The OAAM sample application illustrates the Java API integration and can be downloaded with My Oracle Support article ID 1351899.1 titled *OAAM 11g PS1 (11.1.1.5) Sample Application Download*.

   You can access My Oracle Support at `https://support.oracle.com`.

15.4 OAAM Set Up and Configuration

This chapter includes the following topics:

- Set Up Transaction Definitions
- Set up Policies and Rules
- Sizing and Capacity Requirements

15.4.1 Set Up Transaction Definitions

In order for OAAM to perform risk analysis associated with the client transaction, you must determine how to represent the client transactions in OAAM, how to process the customer data collected by OAAM, and how to use it to prevent fraud and misuse. For example, in an eCommerce transaction, the data to process in the transaction might be credit card numbers, shipping and billing addresses, names, and dollar amounts; for a wire transfer, the important data might be Amount, Name, To account, From account, Routing Number, Bank Address, and Bank Phone. For information on entities, see *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*.

OAAM uses transaction definitions to represent client-specific transactions. Hence, the `External transaction ID` provided by the client should match with the transaction definition key in OAAM.

The transaction definition in OAAM consists of transaction parameters and entities. For information on creating and managing transaction definitions, see *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*

An entity structure is created by combining multiple related data points for optimization. The entity can be reused in multiple transactions by creating new instances of the entity. Entities are not associated with or dependent on any transactions.

For example, `shipping address` and `billing address` instances can be created for two different transactions from the `Address` entity. The address entity can include `street number`, `street name`, `apartment number`, `city`, `state`, `postal code`, and `country` as its data points.

In addition to creating instances, an entity can be also linked to other entities thus establishing a relationship or association between entities. The `Customer` entity can be

linked to another entity like `Address`. The relationship between `Customer` and `Address` entities can be said to be one-to-one (1:1) because they have a one to one direct mapping. The `Address` entity is not dependent on the `Customer` and can reside by itself.

All data fields that cannot be combined to form entities should be added as transaction parameters. Typically these attributes are unique/specific and dependent on the current transaction.

For example:

Transaction Data

- Amount
- Item #

Entities

- Credit Card entity (which includes the data elements like Credit Card Number and Expiry Date)
- Customer (which includes the data elements like first name, last name, date of birth and so on)

15.4.2 Set up Policies and Rules

Oracle Adaptive Access Manager performs risk evaluation and fraud analysis on a client's transaction based on a set of policies and rules defined in OAAM. Follow these guidelines to set up policies and rules for transactions.

1. Determine if you can use any of the out-of-the-box OAAM checkpoints to define new policies for transactions. If an existing checkpoint can be reused, you will not need to create a checkpoint. Otherwise, create a new OAAM checkpoint. For information on creating checkpoints, see Chapter 26, "Creating Checkpoints and Final Actions."
2. Create an OAAM security policy and provide the following information.
   - Name
   - Status
   - Scoring Engine
   - Weight
   - Description
3. Add the rule to the policy. The rule must contain conditions specific to transactions. Provide the following information:
   - Rule Name
   - Status
   - Description
   - Condition
   - Results if the rule condition is satisfied.

     Link the Alert and Action groups and specify a score.

     For details on rule conditions, see *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*.

4. Activate the transaction definitions and policies for OAAM to perform risk evaluation.

   For information on configuring policies, see *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*.

15.4.3 Sizing and Capacity Requirements

For information on sizing and capacity requirements, see *Oracle Fusion Middleware Performance and Tuning Guide*.

This completes the setup and configuration in OAAM.

15.5 Client Setup

To integrate the client application with OAAM, proceed as follows:

1. Write custom client code using the APIs provided by OAAM to create and update transactions and run rules on the transactions. For information on OAAM APIs, see Chapter 4, "Natively Integrating OAAM with Java Applications."
2. Integrate the client application with OAAM using OAAM shared libraries, see Chapter 2, "Natively Integrating Oracle Adaptive Access Manager."
3. Update the client application so it can interpret the results from OAAM and take appropriate action in terms of redirecting the user to the relevant pages. For example, indicate that the user is not allowed to perform wire transfer, and others.

15.6 Entity and Transaction APIs

API	Task
createOAAMSession	Creates a login session.
createTransaction	Creates a transaction in OAAM
updateTransaction	Updates an existing OAAM transaction. Typically this is done based on outcome of processRules API.
processRules	Processes fraud rules
createOrUpdateEntities	Creates or updates entities without sending the full data with the transactions
searchEntityByKey	Searches entities based on its key attributes

15.6.1 Sequence of API Calls

This section shows how transactions are processed in OAAM and supplying the values for API operations from the client application.

- If a Session ID does not exist, call the `CreateOAAMSession` API to create a session containing the Session ID. The Session ID is required by the `createTransaction` API.
- Information is provided by the client application and the `createTransaction` API is called.
- Review to make sure the status of the `createTransaction`operation is `isSuccess()` before obtaining the transaction ID with the method `getTransactionResponse()`.

- Call the `processRules` API to trigger the fraud policies/rules associated to the Transaction checkpoint. This step results in triggering the rules engine that would execute the policies and rules associated to this checkpoint and creating alerts if the associated rules trigger. The output of this API is a set of actions and risk score as returned by the policies and rules.
- Based on the outcome of the `processRules` API call the client application can choose to call the `updateTransaction` API to set the transaction status or to update data in the existing transaction.
- In some cases, client applications can choose to execute a `processRules` API with a `Post Transaction` kind of checkpoint that has policies/rules that have to be executed after a transaction is created.

15.6.2 Out-of-the-Box Checkpoints

The Pre-Transaction and Post-Transaction checkpoints are described in this section.

15.6.2.1 Pre-Transaction Checkpoint

If a create transaction operation was successful, then you can call the `processRules` API to trigger the fraud policies/rules associated to the Pre-transaction checkpoint. This step results in triggering of the rules engine that would execute the policies and rules associated to this checkpoint and creating alerts if the associated rules trigger. The output is a set of actions and risk score as returned by the policies and rules.

15.6.2.2 Post - Transaction Checkpoint

If an update transaction operation was successful, then you can call the processRules API to trigger the fraud policies/rules associated to the Post-transaction checkpoint. This step results in triggering the rules engine that would execute the policies and rules associated to this checkpoint and creating alerts if the associated rules trigger. The output is a set of actions and risk score as returned by the policies and rules.

15.6.3 Entities API List

The two entities APIs are listed as follows.

15.6.3.1 create OrUpdateEntities

You can use the `createOrUpdateEntities` API to perform the following tasks:

- Create and update entities
- Replace and merge attribute values during an entity update

For more information, see Section 4.5.7, "createOrUpdateEntities."

15.6.3.2 SearchEntityByKey

You can use the `searchEntityByKey` API to find entities based on its key attributes. For more information on the API, see Section 4.5.27, "searchEntityByKey."

15.7 Run-time Data Analysis

OAAM provides tools for run-time data analysis of transactions.

15.7.1 Investigation Transaction Search, Comparison, and Utility Panel

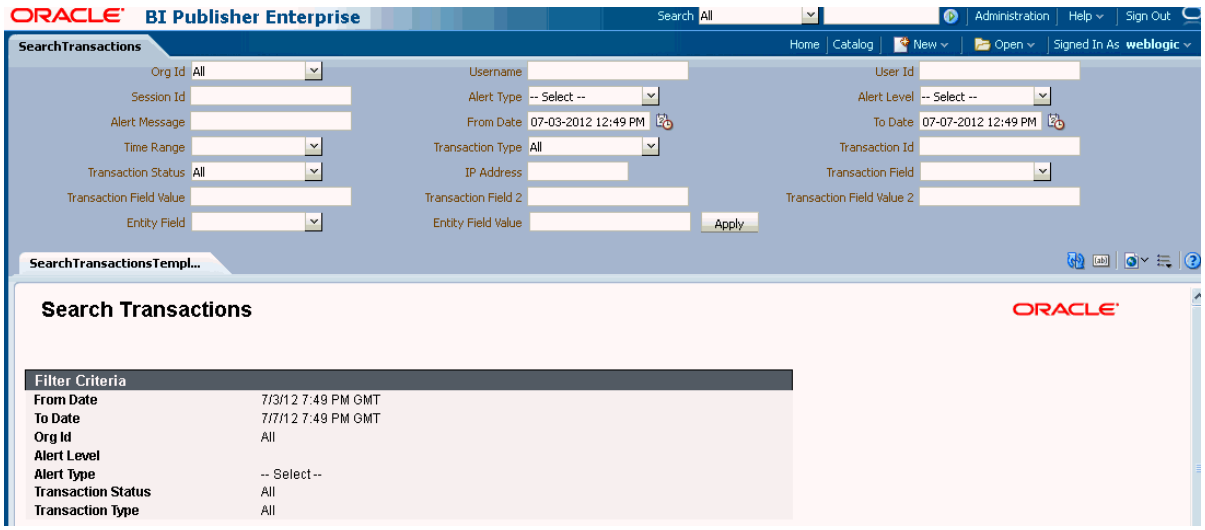
OAAM provides several features for the fraud investigation of transactions, as described in the following table. For details on these investigation tools, see *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*.

Feature	Description
Transaction Search	Investigators can search for OAAM run-time data in a transaction-centric manner using the Sessions and Transactions search pages. For example, the investigator begins the investigation by searching for Retail Ecommerce transactions in the last 24 hours at a certain alert level.
Utility Panel	Investigators can use the Utility Panel to: ▪ Quickly locate sessions and transactions with data in common ▪ Iterate on a query to expand and contract returns ▪ Both view aggregate numbers of sessions and transactions found and drill in to expand investigation
Compare Transactions	Investigators can use the compare transactions feature to compare transactions side by side to find the matching data elements.

15.7.2 BIP Reports

OAAM provides the SearchTransactions report out-of-the-box for transactions. Table 15–2 shows the search filters for the SearchTransactions report.

Figure 15–2 BIP Report



You can also create and configure custom reports on the transaction data as needed. For information on building custom reports, see Section 24.2, "Building OAAM Transactions Reports."

15.8 Targeted Purging of Transaction and Entity Data

To specify a different retention period based on the transaction type or entity, refer to "Setting Up Targeted Purging for Entity Data" and "Setting Targeted Purging for Transaction Data Per Transaction Definition" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*.

16

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 through 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.

This chapter contains the following sections:

16.1 About the OTP 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. The system uses this password to challenge the user to verify 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.

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

16.2 Concepts and Terms

This section provides key definitions, acronyms, and abbreviations that are related to OTP Anywhere Implementation.

Table 16–1 OTP Anywhere Terms

Term	Description
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.
Oracle User Messaging Service (UMS)	The Oracle User Messaging Service is a facility installed in the SOA Domain during installation of the SOA Suite. The Oracle User Messaging Service enables two-way communication between users and deployed applications. The communication can be through various channels, including email, instant messaging (IM or Chat), and SMS. OAAM uses Oracle User Messaging Service as a means of communicating with the user.
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 through SMS and email through Oracle User Messaging Service delivery.
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. You can configure a challenge type 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.

16.3 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 JAR files and other files to an extensions shared library.

For information on adding customized JAR files and other files, see Chapter 7, "Using the OAAM Extensions Shared Library to Customize OAAM."

16.3.1 Install SOA Suite

Before you can configure the Oracle User Messaging Service (UMS) driver and OTP, you must have installed the SOA Suite 11g, configured the SOA Domain and have the Admin Server and the SOA Server running. You also need access to the Oracle Enterprise Manager Fusion Middleware Control Console.

For information on installing the SOA Suite 11g, see *Oracle Fusion Middleware Installation Guide for Oracle SOA Suite and Oracle Business Process Management Suite*.

16.3.2 Configure the Oracle User Messaging Service Driver

The User Messaging Service comes with some drivers that each handle traffic for a specific channel. The drivers control the channels and need to be configured with the properties of the appropriate delivery server, protocol, and so on from which messages are sent. The OAAM Server will be set up for the channels.

Drivers can be deployed or undeployed independently of one another depending on what messaging channels are available in a given installation.

Go to the Oracle Enterprise Manager Fusion Middleware Control Console (typically `http://host:7001/em`) and open the User Messaging Service node. From the drop down menu in the right panel, select the option for the driver properties. For example, choose Email Driver Properties. The form that is shown enables you to set various properties on the driver, including the details about the server (or protocol, and so on) to be used by the driver for the operations. For example, you might enter details about the email server to be used by the driver for email operations.

16.3.2.1 Email Driver

Configure the Email driver to a SMTP server. For information on configuring the Email driver, see *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite*.

Table 16–2 Connecting to the SMTP Server

Parameter	Description
OutgoingMailServer	Mandatory if email sending is required. For example, `smtp.`*`name`*`.com` for *`name`*.
OutgoingMailServerPort	Port number of SMTP server.
OutgoingMailServerSecurity	Possible values are TLS and SSL.
OutgoingDefaultFromAddress (optional)	The email address that is indicated as the sender of the email message.
OutgoingUsername	The user account from which the email is sent.
OutgoingPassword	The account's password (stored in encrypted format).

Press Apply. To have these settings take effect, the driver has to be restarted.

16.3.2.2 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 must 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 16–3 Connecting to the Vendor

Parameter	Description
SmsAccountId	The Account Identifier on the SMS-C. This is your vendor account ID which you must 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 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

After providing the parameter values, press Apply. To have these settings take effect, the driver has to be restarted.

16.4 OTP Setup

OTP using Oracle User Messaging Service (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 JAR files and other files to an extensions shared library. For information on customizing the OAAM server interface, see Chapter 7, "Using the OAAM Extensions Shared Library to Customize OAAM."

Table 16–4 Tasks in OTP Setup

No.	Tasks
1	Install SOA Suite
2	Configure the Oracle User Messaging Service Driver
3	Integrate Oracle User Messaging Service
4	Enable OTP Challenge Types
5	Enable Registration and User Preferences
6	Customize Registration Fields and Validations
7	Customize Terms and Conditions
8	Customize Registration Page Messaging
9	Customize Challenge Page Messaging
10	Customize OTP Message Text
11	Enable Opt Out Functionality
12	Register SMS Processor to Perform Work for Challenge Type
13	Customize One-Time Password Generation
14	Customize One Time Password Expiry Time
15	Configure the Challenge Pads Used for Challenge Types
16	Customize OTP Anywhere Data Storage

The Oracle User Messaging Service (UMS) and 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, see Chapter 2, "Natively Integrating Oracle Adaptive Access Manager."

16.5 Configure OTP

This section contains the following topics:

- Integrate Oracle User Messaging Service
- Enable OTP Challenge Types
- Enable Registration and User Preferences

16.5.1 Integrate Oracle User Messaging Service

The properties to set for the Oracle User Messaging Service (UMS) server URLs and credentials are listed in Table 16–5. 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 Oracle User Messaging Service.

Table 16–5 Oracle User Messaging Service 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 Oracle User Messaging Service server
bharosa.uio.default.ums.integration.password		Password for Oracle User Messaging Service server
bharosa.uio.default.ums.integtaion.policies		Oracle User Messaging Service 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
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 Oracle User Messaging Service server properties, restart the application.

16.5.2 Enable 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 through the challenge type (email, SMS, or IM). The user will see the email, SMS, or IM page in registration flow.

You use the challenge type enum 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 16–6 Oracle User Messaging Service 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

16.5.3 Enable Registration and User Preferences

Enable the registration flow and user preferences by setting these properties to true:

Table 16–7 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.

16.6 Customize OTP

This section contains the following topics:

- Customize Registration Fields and Validations
- Customize Terms and Conditions
- Customize Registration Page Messaging
- Customize Challenge Page Messaging
- Customize OTP Message Text
- Enable Opt Out Functionality

16.6.1 Customize Registration Fields and Validations

Mobile registration field definitions and validations for the OTP registration page are shown in Table 16–8.

Add Mobile Input Registration Field Properties to oaam_custom.properties

These properties should be added to oaam_custom.properties.

Table 16–8 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

Table 16–8 (Cont.) Mobile Input - Properties File

Property	Default Value	Description
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 16–9 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

16.6.2 Customize Terms and Conditions

The following examples show term and conditions definitions for the OTP registration page.

Add Terms and Conditions Definitions to oaam_custom.properties

These properties should be added to oaam_custom.properties.

Table 16–10 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

Table 16–10 (Cont.) Terms and Conditions Checkbox

Property	Default Value	Description
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 16–11 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 popups are listed as follows.

Table 16–12 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

16.6.3 Customize Registration Page Messaging

Add registration properties to client_resource.properties.

Table 16–13 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.

16.6.4 Customize Challenge Page Messaging

Add challenge type fields to client_resource.properties.

Table 16–14 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

16.6.5 Customize OTP Message Text

Add OTP message fields to client_resource.properties.

Table 16–15 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}

16.6.6 Enable Opt Out Functionality

This feature is disabled by default. To enable Opt Out for the user, set the property to true.

Table 16–16 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

16.7 Register SMS Processor to Perform Work for Challenge Type

You use the challenge type enum 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" typically refers to the delivery channel used to send an OTP to the user (Email, SMS, or IM).

Table 16–17 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 in Table 16–18.

Table 16–18 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

16.8 Customize One-Time Password Generation

You can configure the one-time password through properties edits. The following properties are used to generate the OTP:

```
# OTP pin generation config
bharosa.uio.default.otp.generate.code.length = 5
bharosa.uio.default.otp.generate.code.characters = 1234567890
```

The default OTP codes will be 5 characters made up of the numbers 0-9 (for example: 44569).

`bharosa.uio.default.otp.generate.code.length` designates the length of the OTP.

`bharosa.uio.default.otp.generate.code.characters` designates the characters to use when generating the OTP.

An example is shown below for generating a 4 character OTP code with numbers 0-9 and letters a-d (for example: 0c6a):

```
bharosa.uio.default.otp.generate.code.length = 4
bharosa.uio.default.otp.generate.code.characters = 1234567890abcd
```

16.9 Customize One Time Password Expiry Time

To set up OTP SMS password expiry time, add the following property:

`bharosa.uio.default.challenge.type.enum.ChallengeSMS.otpexpirytimeMs`

To set up OTP email password expiry time, add the following property:

`bharosa.uio.default.challenge.type.enum.ChallengeEmail.otpexpirytimeMs` to `oaam_custom.properties`.

The time is in milliseconds. If the value is not in milliseconds, you will have to perform a conversion. For example, to set the expiration time for OTP to be 5 minutes, then you must set the property to `300000 ms` (5 minutes).

Note: This property works for the OTP API, but now OAAM Server does not use the API. Hence, by default, OAAM Server OTP is valid for the session or until used.

16.10 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, 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 16–19 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.

16.11 Customize 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.

16.11.1 com.bharosa.uio.manager.user.UserDataManagerIntf

The methods used in customizations are:

- `public String getUserData(UIOSessionData sessionData, String key);`
- `public void setUserData(UIOSessionData sessionData, String key, String value);`

16.11.2 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. OAAM uses rules to check if a user is registered for a given item and to check the `UserDataFlag` 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.flag.prefix",
                 "otpContactInfoFlag_") + key));
     }

     return false;
   }
```

```
}
```

16.11.3 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.

16.11.4 Configure 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.manager.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.

16.12 Example Configurations

This section contains the following topics:

- Additional Registration Field Definitions Examples
- Additional Challenge Message Examples
- Additional Processors Registration Examples

16.12.1 Additional Registration Field Definitions Examples

Additional registration field definitions are shown in Table 16–20.

Table 16–20 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)

16.12.1.1 Email Input

The following is an example of an enum defining email registration on the OTP registration page of an authenticator:

Table 16–21 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

16.12.1.2 Phone Input

The following is an example of an enum defining phone registration on the OTP registration page of an authenticator:

Table 16–22 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

16.12.1.3 Example - OTP Registration Page to Display Values for Entry of an Email Address Instead of a Mobile Phone

To display only entry information for email and disable entry information for mobile phone for registration, set

bharosa.uio.default.userinfo.inputs.enum.email.enabled=true

```
bharosa.uio.default.userinfo.inputs.enum.mobile.enabled=false
Enabling the email field will require a server restart.
```

16.12.1.4 IM Input

The following is an example of an enum defining IM registration on the OTP registration page of an authenticator:

Table 16–23 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

Table 16–23 (Cont.) IM Input

Property	Default Value	Description
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
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

16.12.2 Additional Challenge Message Examples

Other examples of challenge message resource bundles are in the sections following. These properties must be added to client_resource.properties.

16.12.2.1 Customize OTP Email Message

OTP Email message properties are shown in Table 16–24. Customized OTP email message properties `bharosa.uio.default.ChallengeEmail.message.subject` and `bharosa.uio.default.ChallengeEmail.message.body` must be added to `client_resource.properties`. The property `bharosa.uio.default.ChallengeEmail.message.from.address` must be added to `oaam_custom.properties`.

Table 16–24 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

16.12.2.2 Customize OTP IM Message

OTP IM message properties are shown in Table 16–25.

Table 16–25 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

16.12.3 Additional Processors Registration Examples

Additional processor registration properties are listed in Table 16–26.

Table 16–26 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

16.12.3.1 Register Email Challenge Processor

The properties to register the email challenge processor and mark service as available (or unavailable) are listed in Table 16–27.

Table 16–27 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

16.12.3.2 Register IM Challenge Processor

The properties to register the IM challenge processor and mark service as available (or unavailable) are listed in Table 16–28.

Table 16–28 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

16.13 Challenge Use Case

An example challenge scenario is as follows:

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 through 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.

17

Integrating Mobile Applications with OAAM

This chapter covers the integration of native mobile applications with OAAM. This does not include mobile Web applications that are based on browsers or that use Web views.

This chapter contains the following sections;

17.1 Overview for Integrating Mobile Applications with OAAM

The mobile application integration with OAAM is limited primarily to the server-side integration. The integration developer is responsible for the data exchange between the native mobile application and the business application.

OAAM currently does not provide any software development kit (SDK) that is specific to mobile applications.

> **Note:** If Oracle Access Management Mobile and Social (Mobile and Social) is used, then you do not have to separately integrate the native mobile application with OAAM. Mobile and Social is already integrated with OAAM. For more information, see *Oracle Fusion Middleware Administrator's Guide for Oracle Access Management*.

Table 17–1 lists the high-level steps required to integrate native mobile applications with OAAM.

Table 17–1 Overview of Mobile Application Integration

No.	Step
1	Determine how the mobile device has to be identified. Prepare a list of device specific attributes that can uniquely identify a device. The chapter will refer to this information as mobile device fingerprint.
2	Design and develop client-server interfaces that can send and receive OAAM specific data in addition to the business application data. Typical OAAM specific data includes device identification-related information and session/request ID during the scope of a user session. If transactions are used, then transaction specific data should also be passed accordingly.
3	Write code in the business application to call OAAM APIs to pass the data from the mobile device and also send necessary data from the OAAM API to the mobile device.
4	Test the integration and make sure it is working end-to-end.

The next sections describe the integration tasks in detail.

17.2 Determine Mobile Device Fingerprint

Out of the box OAAM provides a default mobile device fingerprint consisting of the following attributes:

- Operating System Type
- Hardware IMEI Number
- Hardware Mac Address

These are specified through the enum element named `vcrypt.fingerprint.type.enum.native_mobile`. You can use the Properties editor option of OAAM Admin Console to view the attributes of this enum element.

To modify the mobile fingerprint:

1. Log in to the OAAM Admin Console.
2. In the Navigation pane, double-click **Properties** under the **Environment** node. The **Properties Search** page is displayed.
3. Enter `vcrypt.fingerprint.type.enum.native_mobile` in the **Name** field and click **Search**.

   You should see the attributes of the property in the Search Results section.
4. If you must add more attributes that identify your mobile device, enter `vcrypt.fingerprint.type.enum.native_mobile.header_list` in the **Name** field and click **Search**.
5. Click to select the property in the Search Results section, add the list of attributes and click **Save**.
6. To provide the mapping of attributes to their display value, enter `vcrypt.fingerprint.type.enum.native_mobile.header_name_nv` in the **Name** field and click **Search**.
7. Click to select the property in the Search Results section, add the mapping attributes, and click **Save**.
8. To provide the mapping of actual value to the display value, enter `vcrypt.fingerprint.type.enum.native_mobile.header_value_nv` in the **Name** field and click **Search**.

9. Click to select the property in the Search Results section, add the actual value to display, and click **Save**.

Note: Do not add attributes that are very dynamic in nature like IP Address, and so on, as mobile device fingerprint attributes. Add only those that can uniquely identify the device.

The following example shows a mobile fingerprint enum:

```
vcrypt.fingerprint.type.enum.native_mobile=900
vcrypt.fingerprint.type.enum.native_mobile.name=Native Mobile
vcrypt.fingerprint.type.enum.native_mobile.description=
Native Mobile implementation using Mobile and Social
vcrypt.fingerprint.type.enum.native_mobile.processor=
     com.bharosa.uio.processor.device.NativeMobileDeviceIdentificationProcessor
vcrypt.fingerprint.type.enum.native_mobile.header_list=
      os.type,os.version,hw.imei,hw.mac_addr
vcrypt.fingerprint.type.enum.native_mobile.header_name_nv=
       os.type,Operating System Type,os.version,Operating System Version,
       hw.imei,Hardware IMEI Number,hw.mac_addr,Hardware Mac Address
vcrypt.fingerprint.type.enum.native_mobile.header_value_nv=t,true,f,false
```

17.3 Develop/Enhance Client Server Interfaces to Handle OAAM-Specific Data

Certain OAAM-specific data must be passed from the native mobile application (client) to OAAM and vice versa. To achieve this, you will need to design, develop, and enhance the client server interfaces so that they can handle OAAM-specific data.

The following are the typical OAAM data that must be handled:

OAAM Device Fingerprint Data

- Client (native mobile application)

  Send all the device identification related data to the server.

- Server
  - Parse the device identification related data and format device fingerprint.
  - Use the device fingerprint data while OAAM Session creation API Calls.

OAAM Device Cookies

- Client (native mobile application)
  - Pass existing OAAM cookies to the server
  - Persist new OAAM cookies that server returns
  - Store cookies so that they are secure. (They cannot be stolen)
- Server
  - Get existing cookies from the native mobile application (client) and use them in the OAAM Session creation APIs
  - Pass the new device cookies to client

Pass Client Application Id/Name

- Client should send the mobile application name

- Server should use the mobile application name as `Client Application` when calling OAAM Session Creation API

OAAM Session/Request-Id

- Client (native mobile application)

  Generate new or use existing OAAM Session/Request Id based on whether it is a new session or an existing session. Care must be taken to dispose the OAAM Session ID as soon as the session ends/expires.

- Server

  Use the Session/Request Id from the client appropriately in the calls to OAAM API

Handling Transactions data

- Client (native mobile application)

  Send the data required to identify the type of transaction and related data

- Server
  - Determine the transaction definition key and transaction context data based on the data from client
  - Use the transaction definition key and transaction context data and call OAAM Transaction APIs
  - Call the transaction related checkpoints/rules and send the result to client

17.4 Out-of-the-box Mobile Device Identification Policy

This section describes how the out-of-the-box mobile device identification policy works.

17.4.1 Identify Device by Mobile Cookie

Mobile Cookie identifies the device if the following conditions are satisfied:

- Mobile cookie is valid
- There is no mismatch with known headers

In this case the score would be zero (0)

17.4.2 New Device

Device is treated as New Device if any of the following conditions are satisfied:

- Mobile cookie is valid but the known headers do not match
- Mobile cookie is invalid (or stale)
- If both the first two are not true then a background check is performed to see if the device can be identified using the mobile device fingerprint

17.5 Review Out-of-the-Box Security Policies and Develop Custom Policies If Required

Review the following rules in OAAM Post Authentication Security policy that are specific to mobile devices:

Lost or Stolen Device

Detects if the current device is reported as lost or stolen

Rule	Rule Condition and Parameter Values	Results
Lost or Stolen Device	Device: Check if device is of given type Device Type = Mobile Device Return Value = True Device: Device in group Is in group = True Device in group = OAAM Lost or stolen Device	Action = OAAM Lost Device Alert = OAAM Lost or Stolen Device Score = 1000

Jailbroken Mobile Device

Detects if the current device is jail broken based on the context data from mobile device.

Rule	Rule Condition and Parameter Values	Results
Jail broken Mobile Device	**Device: Check if device is of given type** Device Type = Mobile Device Return Value = True **Session: Check string parameter value** Parameter Key = isJailBroken Value = true	Action = OAAM Challenge Alert = OAAM Jailbroken Device Score = 500

Too many mobile devices

Detects if the user is logging in from too many unregistered mobile devices

Rule	Rule Condition and Parameter Values	Results
Too many mobile devices	**Device: Check if device is of given type** Device Type = Mobile Device Return Value = True **DEVICE: Is registered** Is Registered then return = False **User: Check Number of Registered Devices of a Given Type** Number Of Devices = More than Number Of Devices to compare = 4 Device Of Type = Mobile Device	Action = OAAM Too Many Mobile Devices Alert = OAAM More Mobile devices used than allowed Score = 1000

Black Listed Mobile Devices

Detects if the current login is from a black-listed device

Hardware Identifier same but Operating System mismatch

Detects if the HW Identifier is same as hardware identifier of a previously identified device but the Operating System is not matching.

Rule	Rule Condition and Parameter Values	Results
Hardware Identifier same but Operating System mismatch	Precondition: Device Risk Score between 599 and 601 **Device: Check if device is of given type** Device Type = Mobile Device Return Value = True **Device: Browser Header Substring** Substring = "OIC"	Action = OAAM Mobile Device OS Mismatch Alert = OAAM Mobile Device with Different OS Score = 1000

For any additional/custom requirements the recommendation is to create another Policy and add the required rules to that policy.

17.6 Process to Manage Lost or Stolen Devices

To detect lost or stolen devices, the related lists have to be populated with the device IDs that are reported as lost or stolen.

The following is the recommended approach:

1. Log in to OAAM Admin Console
2. Identify the Device ID of the device that is reported as lost or stolen.
   a. This can be done searching for the user sessions and then narrowing down to sessions that can be confirmed where the device was used
   b. Note the Device ID of the related device
3. Search for Groups of type `Devices`.
4. Select the **OAAM Lost or Stolen Devices** group
5. Add the Device ID to the group.

17.7 Process to Manage Black Listed Devices

The process is very similar to the process to manage "Lost or Stolen" Devices. The difference is adding the Device Id to the `OAAM Black listed mobile devices` group.

17.8 Handle Mobile Specific Rule Outcomes

Out of the box OAAM Post Authentication Policy could return the following outcomes that are specific to mobile devices. These should be appropriately handled for the integration code to take business specific actions:

- OAAM Lost Device
- OAAM Too Many Mobile Devices
- OAAM Black listed Mobile Device
- OAAM Mobile Device OS mismatch

17.9 Customizing User Interface for Mobile Devices

The OAAM Server can be customized to be mobile friendly. A base mobile CSS (external stylesheet) file has been provided to enable users to view the Web pages on mobile devices. Because significant differences exist between the browsers found on devices supported by OAAM and the native look-and-feel of each device varies greatly, OAAM Server supports a custom override to the mobile CSS. Device-specific CSS can be defined in addition to generic mobile CSS to allow for more fine grade customizations. Mobile devices include iPhone, iPad, Android, Windows Phone, and Blackberry.

To implement a device specific CSS file:

1. Make the external CSS file.

   For example, `customAndroid.css`, `customIPhone.css`, `customBlackBerry.css`, `customWindows7phone.css`, and so on.

2. Define all the style rules within this CSS text file. For example, add a style that makes all body text italic:

   ```
   body{
   font-style:italic
   }
   ```

3. Add the following reference to the custom file in `oaam_custom.properties`:

   ```
   bharosa.uio.default.custom.mobile.css=/css/customMobile.css
   ```

   Add custom mobile CSS per mobile device type by including the name of the device in the property, so that

   bharosa.uio.default.custom.mobile.*device*.css=/css/custom*device*.css

   device can be any defined device, such as iPhone, Blackberry, Android, and Windows Phone.

   ```
   bharosa.uio.default.custom.mobile.android.css=/css/customAndroid.css
   bharosa.uio.default.custom.mobile.iphone.css=/css/customIPhone.css
   bharosa.uio.default.custom.mobile.blackberry.css=/css/customBlackBerry.css
   bharosa.uio.default.custom.mobile.windowsphone7.css=
   /css/customWindowsPhone7.css
   ```

4. Add the CSS file in the OAAM Extensions library.

17.10 Custom Mobile CSS File Inclusion

Custom mobile CSS file inclusion depends on definition of the property and inclusion of the CSS file in the OAAM Extensions library.

The default inclusion order is as follows:

1. Product CSS
2. Custom CSS (if defined)
3. Product right to left CSS (if right to left locale)
4. Custom right to left CSS (if right to left locale and defined)
5. Product mobile CSS (if mobile device)
6. Custom mobile CSS (if mobile device and defined)

7. Product mobile device specific CSS (if defined for device)
8. Custom mobile device specific CSS (if defined for device)

If files define the same attribute, the file that comes later overrides the earlier one.

In additional to the custom CSS file, the OAAM out of the box policies provide for an the HTML version of the question registration page for a user on a mobile device.

- The Authentication pad policy returns the `HTML` action when a user is required to register new security questions and the user is accessing the application from a mobile device web browser.
- The Authentication pad policy returns the `Question Pad` action when user is required to register a new security questions and the user is accessing the application from a non-mobile device web browser.

18

Integrating Juniper Networks Secure Access (SA) and OAAM

The integration of Juniper Networks Secure Access (SA) and Oracle Adaptive Access Manager provides enterprises with a remote access control solution with strong multi-factor authentication and advanced real time fraud prevention capabilities to enable secure access to an enterprise's applications.

This chapter explains how to configure OAAM for integration with Juniper Secure Access (SA). This chapter contains the following sections:

18.1 Introduction

To access a protected enterprise resource in the network security trust zone, the user must access SSL VPN, which is the secured gateway for any remote access.

Juniper SA is a series of SSL VPN appliances that ensure that remote and mobile employees, customers, and partners have secure anytime, anywhere access to corporate resources and applications.

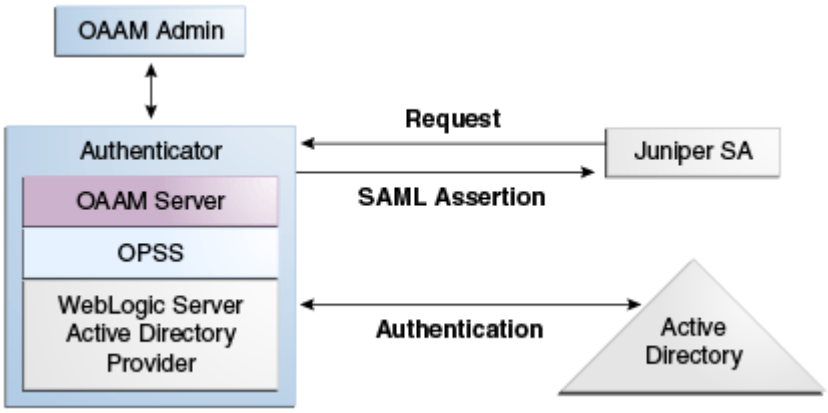
Oracle Adaptive Access Manager 11*g* safeguards vital online business applications with strong yet easily deployed risk-based authentication, anti-phishing, and anti-malware capabilities.

SAML (Security Assertion Markup Language) is an XML-based open standard for exchanging authentication and authorization data between security domains.

In this integration, Juniper SA, which controls access to resources, uses OAAM during authentication to minimize risk and enhance security during the authentication of the user. The combined solutions enable the detection of fraud and risk during authentication and accordingly strongly authenticate the user using OAAM capabilities like `Challenge`, `Block`, and other actions. Juniper SA is configured to use Security Assertion Markup Language (SAML) to exchange user authentication and authorization data.

Figure 18–1 shows the high level flow between Juniper SA and OAAM. For information on the authentication flow, see Section 18.2.1, "Authentication Flow."

Figure 18–1 Juniper SSL VPN and OAAM Integration Architecture



18.2 Authentication and Forgot Password Flows

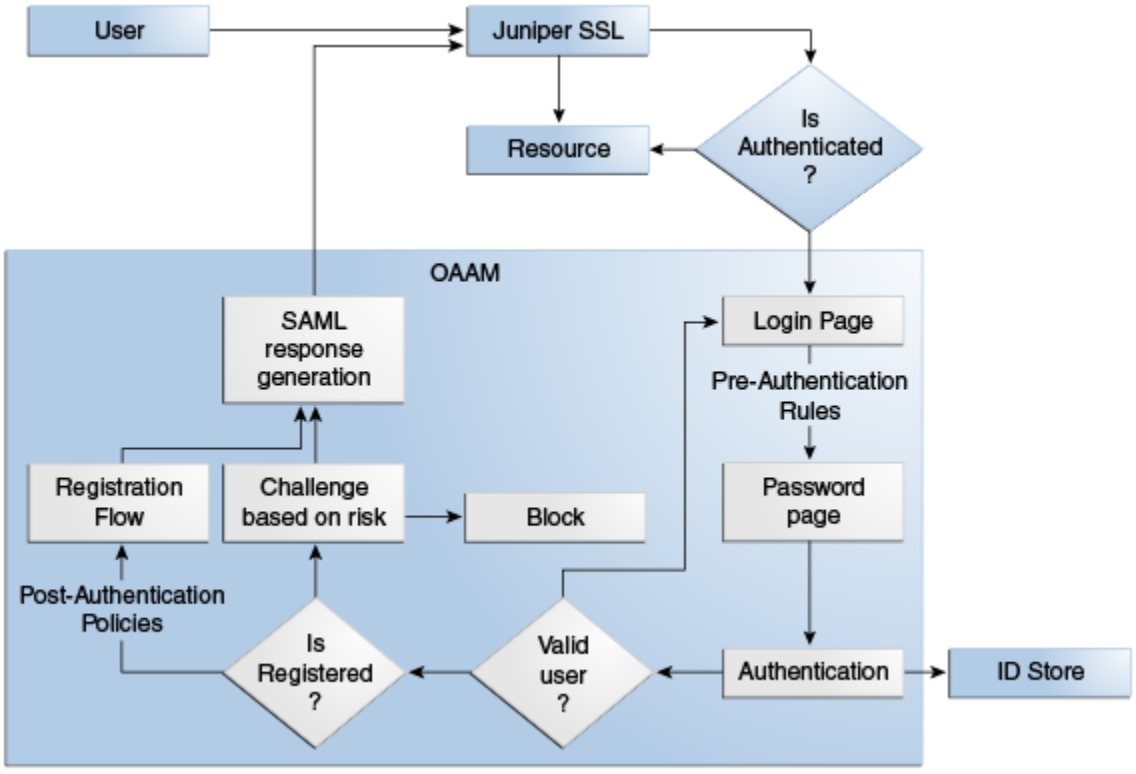
The two integration use cases focus on integrating OAAM for authentication and forgot password flows in Juniper SSL VPN.

- OAAM detects fraud and risk during authentication and provides strong authentication capabilities like `Challenge`, `Block`, or other actions. The integration redirects the user to OAAM to fulfill the authentication sequence if the user is not authenticated.
- OAAM provides password reset authentication. The Forgot Password flow allows users to reset their password after successfully answering challenge questions.

18.2.1 Authentication Flow

Figure 18–2 shows how a user logs into a Web application or URL that is secured by Juniper SA with OAAM providing the authentication flow.

Figure 18–2 Juniper SA with OAAM Providing the Authentication Flow



The following process explains how a user logs into a Web application or URL that is secured by Juniper SA with OAAM providing the authentication flow.

1. The user tries to access a Web application or URL that is secured by Juniper SA. Juniper SA is configured to use SAML.
2. Juniper SA detects whether the user is authenticated or not. If authenticated, the user is allowed to proceed to the Web application.
3. If not authenticated, the user is redirected to the OAAM Server. The OAAM Server displays the OAAM Login page and prompts the user to enter the User ID.
4. Once the user enters the User ID, as part of credential collection, OAAM evaluates the Pre-Authentication checkpoint to verify if the user has to be blocked. OAAM then checks to see if the user has registered for the Authentication Pad. If so, OAAM displays the registered Authentication Pad; otherwise, OAAM displays a generic text pad.
5. The OAAM Server displays the Password page with the Authentication Pad and prompts the user to enter the password.
6. Once the password is entered, OAAM uses Oracle Platform Security Services (OPSS) to validate it against the user store (the user store can be LDAP, Active Directory, or other authentication provider). OPSS is a standard-based, portable, integrated, enterprise grade security platform for Java applications.

   OAAM also identifies the device by running the device identification process.
7. If the credentials are incorrect, OAAM displays an error page and asks the user to enter the credentials again.

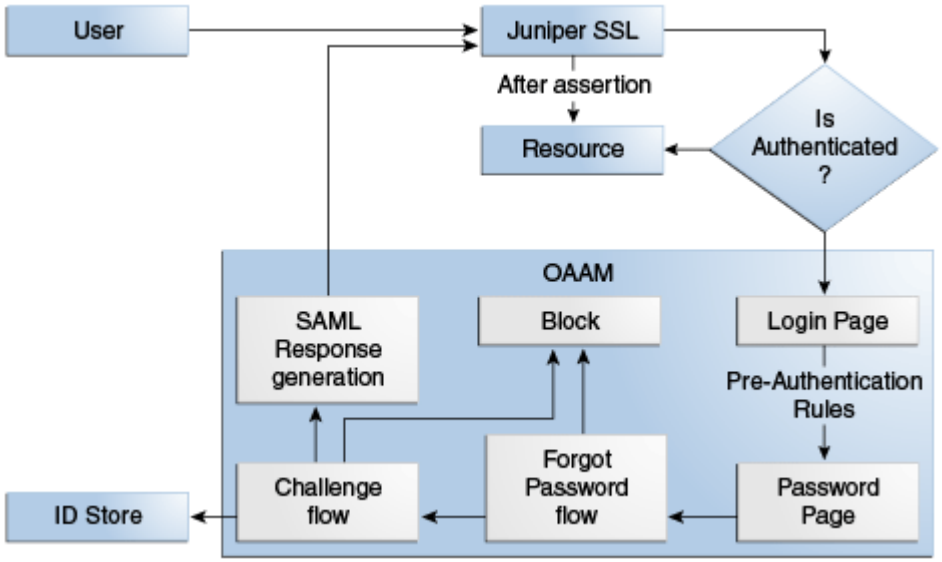
8. If the credentials are correct, OAAM evaluates the Post-Authentication checkpoint. Based on the outcome of the checkpoint, OAAM challenges or blocks the user.
9. If the outcome of Post-Authentication is `ALLOW`, then OAAM determines if the user has to be registered. Based on the types of registration, it takes the user through the registration pages.

   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.
10. If the outcome of Post-Authentication is `CHALLENGE` and if the user is already registered for at least one of the challenge mechanisms, OAAM challenges the user. If able to answer the challenge, the user is allowed to the next step.
11. After successful authentication, OAAM gets the user attributes from the user store and then creates the SAML assertion based on the user attributes from the user store, signs it and then posts the assertion to the Juniper SA redirection URL. Juniper SA consumes and validates the assertion and logs the user to the user requested target page or web application.
12. If the outcome of the Post-Authentication checkpoint is `BLOCK`, then the user is blocked and will not be able to access the web application that the user tried to access.

18.2.2 Forgot Password Flow

Figure 18–3 shows how a user resets the password after successfully answering all challenge questions.

Figure 18–3 Juniper SA with OAAM Forgot Password Flow



The following process explains how a user resets the password after successfully answering all challenge questions.

This use case focuses on integrating OAAM for the Forgot Password flow with Juniper SA.

1. The user tries to access a Web application or URL that is secured by Juniper SA. Juniper SA is configured to use SAML.
2. Juniper SA detects whether the user is authenticated or not. If authenticated, then the user is allowed to proceed to the web application.
3. If not authenticated, the user is redirected to the OAAM Server.
4. The OAAM Server displays the OAAM Login page and prompts the user to enter the User ID.
5. Once the user enters the User ID, OAAM evaluates the Pre-Authentication checkpoint and checks to see if the user has to be blocked. OAAM then checks to see if the user has registered for the Authentication Pad. If so, it displays the registered Authentication Pad; otherwise, it displays the generic text pad.
6. The OAAM Server display the Password page with the Authentication Pad and prompts the user to enter the password.
7. The user clicks the **Forgot Password** link.
8. The OAAM Server starts the Forgot Password flow by running the Forgot Password checkpoint.
9. Based on the outcome, OAAM Server either challenges or blocks the user.
10. If able to successfully answered the challenge, the user is prompted to enter the new password.
11. Then OAAM makes a call to the user store to update the password.
12. It then gets the user attributes from the user store and then creates the SAML assertion, signs it and then posts the assertion to the Juniper SA redirection URL.

18.3 Security and Authentication Integration

To integrate Oracle Adaptive Access Manager and Juniper Networks Secure Access (SA) to use Oracle Adaptive Access Manager's Authentication and Forgot Password flows, refer to procedures in this section.

18.3.1 Integration Roadmap

Table 18–1 lists a summary of the high-level tasks for integrating Oracle Adaptive Access Manager and Juniper SA.

Table 18–1 Integration Steps

No.	Task	Information
1	Review pre-requisites.	For information, refer to Pre-requisites.
2	Configure the authentication provider.	For information, refer to Configure the Authentication Provider.
3	Configure Oracle Platform Security Services for authentication.	For information, refer to Configure Oracle Platform Security Services (OPSS) for Integration.
4	Import server properties.	For information, refer to Import the SAML Configuration-Related Server Properties Using the OAAM Administration Console.

Table 18–1 (Cont.) Integration Steps

No.	Task	Information
5	Set up Certificate of Trust.	For information, refer to Set Up Certificate for Signing the Assertion.
6	Modify integration properties.	For information, refer to Modify Integration Properties Using the OAAM Administration Console.
7	Configure Juniper SSL	For information, refer to Configure Juniper Networks Secure Access (SA).

18.3.2 Pre-requisites

Before starting tasks in this chapter, be aware that:

- Synchronizing system clocks on Juniper SA and OAAM Servers nodes is required. The time must be synchronized between the Juniper SA system clock and the OAAM Server clock. The time on the Juniper SA system clock must not be ahead of the OAAM Server clock.
- Oracle WebLogic is required. Consult the current certification matrix for the WebLogic version. Currently this integration is tested on OAAM 11*g* on Oracle WebLogic only.
- This integration is implemented in SAML 1.1 version.

18.3.3 Configure the Authentication Provider

In WebLogic, Authentication providers are used to prove the identity of users or system processes. Authentication providers also remember, transport, and make that identity information available to various components of a system (through subjects) when needed.

You can use WebLogic's embedded LDAP, AD (Active directory), and other authentication providers as the Identity Store.

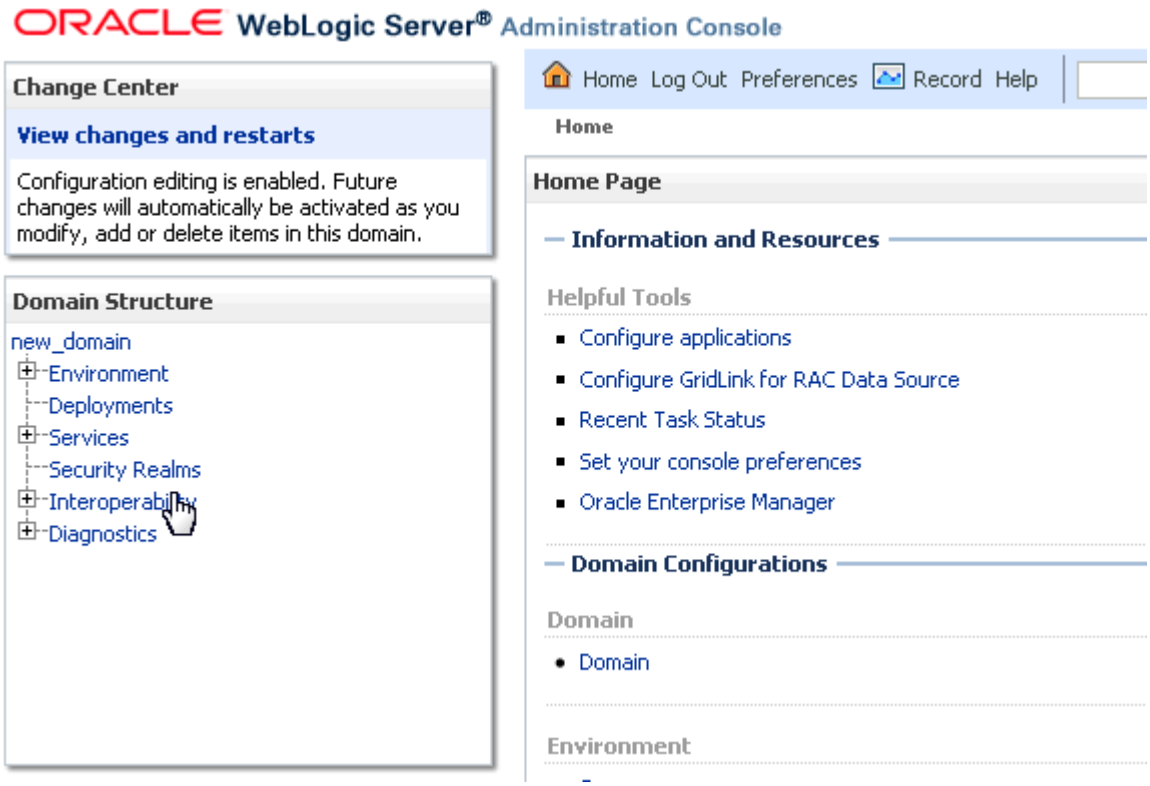
To configure an Authentication provider:

1. Log in to the Oracle WebLogic Administration Console as a WebLogic administrator. For example:

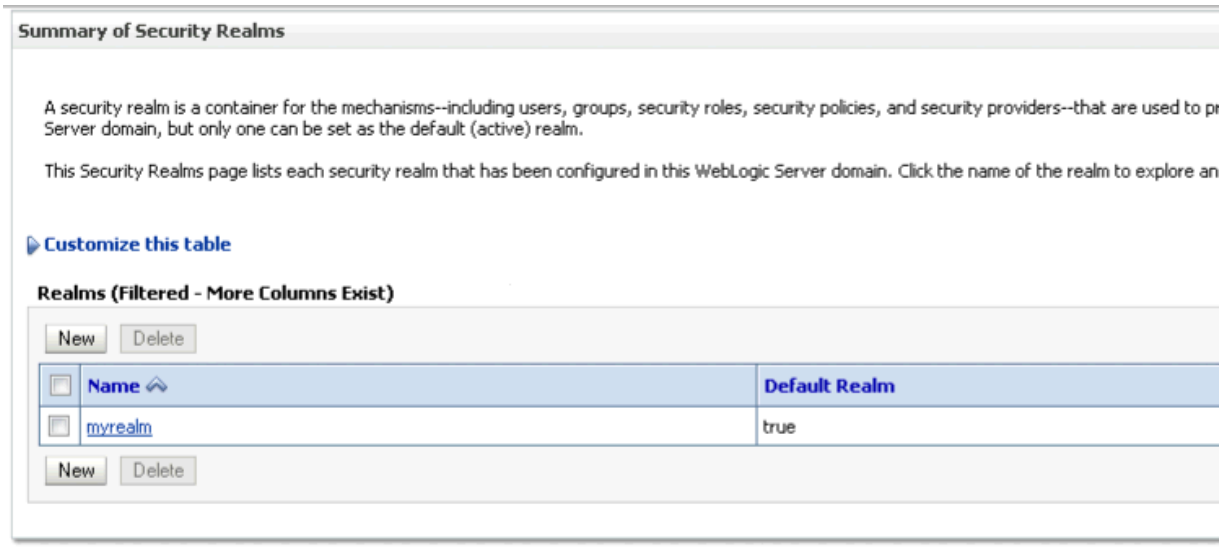
   `http://hostname:7001/console`

   The default port is `7001`.

2. Select **Security Realms** from the **Domain Structure** section in the left pane.

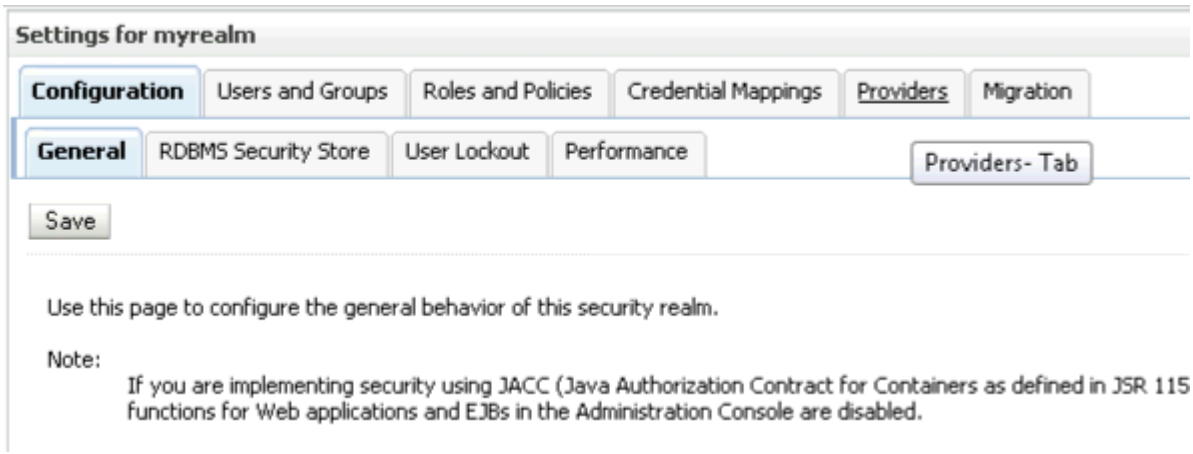


3. Click the name of the realm you are configuring in the Summary of Security Realms page. For example, `myrealm`.



The Settings for myrealm page is displayed.

4. Click the **Providers** tab to display the Authentication subtab.



5. Select **DefaultAuthenticator** to use embedded LDAP. For other Identity stores choose the appropriate provider. For example, **AD Authenticator** for Active directory.

6. Reorder the providers so that **DefaultAuthenticator** is in the first position.
   a. Click **Reorder** to display the Reorder Authentication Providers page.
   b. Select **DefaultAuthenticator** and use the arrow buttons to move it into the first position in the list.
   c. Click **OK** to save your changes.
7. If you are using the embedded LDAP, you must perform the following steps:
   a. In the Oracle WebLogic Administration Console, click **base_domain** in **Domain Structure** in the left pane.
   b. Click the **Security** tab.
   c. Click the **Embedded LDA**P tab.
   d. Select the **Master First** option.

e. Click **Save**.

8. Restart the WebLogic Administration server and the managed servers.

18.3.4 Configure Oracle Platform Security Services (OPSS) for Integration

Oracle Platform Security Services (OPSS) provides enterprise product development teams, systems integrators (SIs), and independent software vendors (ISVs) with a standards-based, portable, integrated, enterprise-grade security framework for Java Standard Edition (Java SE) and Java Enterprise Edition (Java EE) applications. The OPSS abstraction layer is used for authentication. The authentication configuration must be performed in WebLogic. For information on OPSS, see *Oracle Fusion Middleware Application Security Guide*.

1. On the machine where OAAM is installed, navigate to *WebLogic_Domain*/config/fmwconfig.
2. Back up jps-config.xml.
3. Open the jps config.xml.
4. Before closing tag </jpsContexts> add the following jps context:

```
<!-- This context is used for OAAM Juniper Integration -->
<jpsContext name="idcontext">
<serviceInstanceRef ref="user.authentication.loginmodule"/>
<serviceInstanceRef ref="idstore.ldap"/>
<serviceInstanceRef ref="credstore"/>
<serviceInstanceRef ref="keystore"/>
<serviceInstanceRef ref="policystore.xml"/>
<serviceInstanceRef ref="audit"/>
</jpsContext>
```

5. Save the file and exit.

> **Note:** Once you save the file, you might want to use an XML editor to check that all the tags are correct. You can also open the file in Internet Explorer to see whether there are any tags missing. If your changes are correct you will be able to open the file successfully in Internet Explorer.

6. Stop and start the WebLogic Administration Server, OAAM Admin Server and OAAM Managed Server, since these changes requires a restart.

18.3.5 Import the SAML Configuration-Related Server Properties Using the OAAM Administration Console

Import the SAML configuration-related properties so they are added in the OAAM database.

To import the SAML configuration-related properties, proceed as follows:

1. Log in to the OAAM Administration Console as a security administrator. For example:

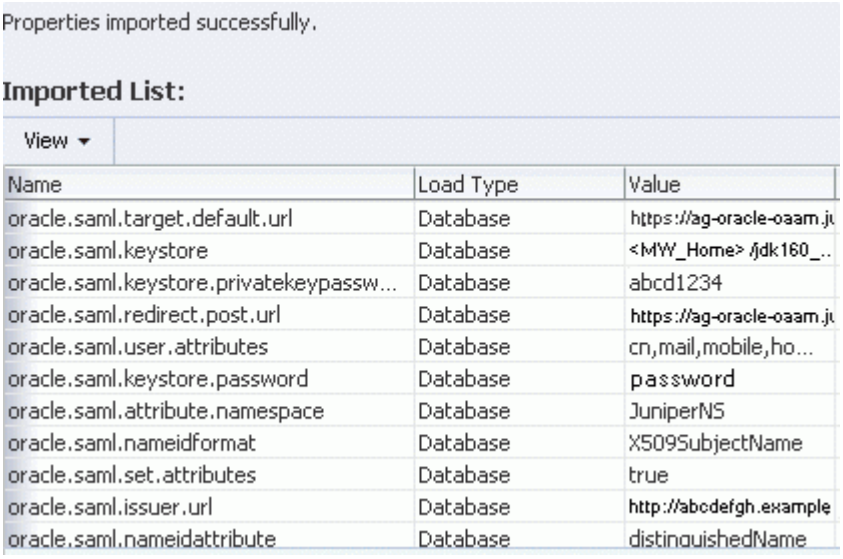
   http://*hostname*:port/oaam_admin

2. In the Navigation pane, click **Properties** under the **Environment** node.

3. Click **Import Properties** in the Properties page to import server properties for the integration

4. Browse for `saml_properties.zip` in the *`IDM_ORACLE_HOME`*`/oaam/init` directory, and click **Open**, and then, click **Import**.

   Once the import is complete it will show you the properties successfully imported.



5. Click **Done** to complete the import.

   This will import the properties needed for the integration. You will modify these properties according to your environment in Section 18.3.7, "Modify Integration Properties Using the OAAM Administration Console."

18.3.6 Set Up Certificate for Signing the Assertion

A certificate authority (CA) is a trusted third-party that certifies the identity of third-parties and other entities, such as users, databases, administrators, clients, and servers. The certificate authority verifies the party identity and grants a certificate, signing it with its private key.

To set up the certificate of trust between Juniper SA and OAAM, follow the procedures contained in these sections:

- Create Private Key for Certificate
- Create a Certificate Request
- Act as Your Own Certificate Authority

- Import the Certificate into Your Keystore

18.3.6.1 Create Private Key for Certificate

The first step is to create a private key for the certificate. To create this private key, proceed as follows:

1. Change the working directory to the security properties directory *MW_HOME*/jdk160_18/jre/lib/security.
2. Create the private key using a key and certificate management utility, called keytool. Enter the following command with cacerts as the keystore:

```
keytool -genkey -keyalg rsa -validity 1825
        -keysize 2048 -alias OAAMCert
        -keystore cacerts -storepass changeit
```

3. Enter the details for the certificate.

   An example of the output is shown as follows:

```
What is your first and last name?
  [Unknown]:  ag-oracle-oaam
What is the name of your organizational unit?
  [Unknown]: Juniper
What is the name of your organization?
  [Unknown]: Juniper
What is the name of your City or Locality?
  [Unknown]:  Sunnyvale
What is the name of your State or Province?
  [Unknown]:  CA
What is the two-letter country code for this unit?
  [Unknown]:  US
Is CN=ag-oracle-oaam, OU=Juniper, O=Juniper, L=Sunnyvale, ST=CA, C=US correct?
  [no]:  yes
```

> **Note:** Typically the CN of the certificate is the name of the machine.

4. When prompted, enter the keystore password:

```
Enter key password for <OAAMCert>
        (RETURN if same as keystore password):
Reenter new password:
```

   Remember this password as it is needed later for the integration.

18.3.6.2 Create a Certificate Request

After you create the private key and self-signed certificate, use the keytool command to generate a Certificate Signing Request (CSR):

1. Change the working directory to *MW_HOME*/jdk160_18/jre/lib/security.
2. Run this command to create the certificate request:

```
keytool -certreq -alias OAAMCert -file server.csr
-keystore cacerts -storepass changeit
```

   In this example you created a certificate request in a file called server.csr.

18.3.6.3 Submit the Certificate Signing Request (CSR) to a Certificate Authority

Submit the Certificate Signing Request (CSR) to a certificate authority to obtain a digital certificate. The certificate authority will issue the certificate. You would have to receive the issued certificate and the root CA Certificate which signed the request.

For testing, you can act as your own certificate authority to sign the certificates. For production scenarios, a certificate from a certificate authority has to be used.

For production scenarios, you can skip Section 18.3.6.4, "Act as Your Own Certificate Authority" and go to Section 18.3.6.5, "Import the Certificate into Your Keystore" to import the certificate from an external certificate authority.

18.3.6.4 Act as Your Own Certificate Authority

For testing purposes, you can act as your own certificate authority to self-sign the certificates. The following sets of instructions walk through setting up to self-sign the certificates. To set this up, proceed with the subsequent example.

18.3.6.4.1 Prerequisites The package `OpenSSL` must be installed in the machine you will use to manage your certificates or create the certificate requests. OpenSSL is an open source implementation of the Secure Sockets Layer (SSL) protocol. OpenSSL implements basic cryptographic functions and provides utility functions.

18.3.6.4.2 Create the Necessary Directories To create the necessary directories, proceed as follows:

1. Create a directory where all certificate files will be kept. The default directory is `/etc/pki/tls/`. As root, issue the following command to create your own directories:

   ```
   # mkdir -m 0755 /etc/pki_jungle
   ```

2. Then, create the certificate authority's directory by issuing the commands:

   ```
   # mkdir -m 0755 \
   /etc/pki_jungle/myCA \
   /etc/pki_jungle/myCA/private \
   /etc/pki_jungle/myCA/certs \
   /etc/pki_jungle/myCA/newcerts \
   /etc/pki_jungle/myCA/crl
   ```

where

- `myCA` is your certificate authority's directory
- `myCA/private` is the directory where your private keys are placed. Be sure that you set restrictive permissions to all your private keys so that they can be read only by root, or the user with whose privileges a server runs. The consequences of a certificate authority private key being stolen would be catastrophic.
- `myCA/certs` directory is where your server certificates will be placed
- `myCA/newcerts` directory is where OpenSSL puts the created certificates in PEM (unencrypted) format and in the form `cert_serial_number.pem` (For example: `07.pem`). OpenSSL needs this directory, so you must create it
- `myCA/crl` is where your certificate revocation list is placed

18.3.6.4.3 Initial OpenSSL configuration

1. To copy the default OpenSSL configuration file (`openssl.cnf`) from `/etc/pki/tls` to your certificate authority's directory and name it `openssl.my.cnf`, issue the following command as root:

   ```
   # cp /etc/pki/tls/openssl.cnf /etc/pki_jungle/myCA/openssl.my.cnf
   ```

2. Since this file does not need to be world readable, you can change its attributes by issuing the command:

   ```
   # chmod 0600 /etc/pki_jungle/myCA/openssl.my.cnf
   ```

3. Create the file that serves as a database for OpenSSL, by issuing the command:

   ```
   # touch /etc/pki_jungle/myCA/index.txt
   ```

4. Create the file which contains the next certificate's serial number, by issuing the command:

   ```
   # echo '01' > /etc/pki_jungle/myCA/serial
   ```

   Since you have not created any certificates yet, set it to "`01`":

18.3.6.4.4 Create the CA Certificate and Private Key After completing the initial configuration, you can now generate a self-signed certificate that will be used as your certificate authority's certificate to sign other certificate requests and a private key.

1. Change to your certificate authority's directory.

   As root, issue the OpenSSL command:

   ```
   # cd  /etc/pki_jungle/myCA/
   ```

   This is where you should issue all OpenSSL commands since it is the location of your OpenSSL's configuration file (`openssl.my.cnf`).

2. Then, create your certificate authority's certificate and private key. As root, issue the following command:

   ```
   # openssl req -config openssl.my.cnf -new -x509 -extensions v3_ca
   -keyout private/myca.key -out certs/myca.crt -days 1825
   ```

   This creates a self-signed certificate with the default CA extensions which are valid for five years.

3. When prompted for a passphrase for your certificate authority's private key, set a strong passphrase.

4. When prompted, provide information that will be incorporated into your certificate request. Information for the certificate authority is similar to the example that is shown:

   ```
   Country Name (2 letter code) [GB]:GR
   State or Province Name (full name) [Berkshire]:Greece
   Locality Name (For example, city) [Newbury]:Thessaloniki
   Organization Name (For example, company) [My Company Ltd]:My Network
   Organizational Unit Name (For example, section) []:My Certificate Authority
   Common Name (For example, your name or your server's hostname)
   []:server.example.com
   Email Address []:whatever@server.example.com
   ```

   Two files are created:

`certs/myca.crt`: This is your certificate authority's certificate and can be publicly available and world readable.

`private/myca.key`: This is your certificate authority's private key. Although it is protected with a passphrase, you should restrict access to it so that only root can read it.

5. Although your certificate authority's private key is protected with a passphrase, you should restrict access to it so that only root can read it. To do so, issue the following command:

```
# chmod 0400 /etc/pki_jungle/myCA/private/myca.key
```

18.3.6.4.5 More OpenSSL Configuration (Mandatory) Modifications to `/etc/pki_jungle/myCA/openssl.my.cnf` are necessary because you use a custom directory for your certificates' management.

1. Open `openssl.my.cnf` in a text editor as root and find the following section (around line 35):

```
[ CA_default ]
dir                = ../../CA              # Where everything is kept
certs              = $dir/certs            # Where the issued certs are kept
crl_dir            = $dir/crl              # Where the issued crl are kept
database           = $dir/index.txt        # database index file.
#unique_subject = no                       # Set to 'no' to allow
                                           # creation of
                                           # several certificates with
                                           # the same subject.
new_certs_dir      = $dir/newcerts         # default place for new certs.
certificate        = $dir/cacert.pem       # The CA certificate
serial             = $dir/serial           # The current serial number
#crlnumber         = $dir/crlnumber        # the current crl number must be
                                           # commented out to leave a V1 CRL
crl                = $dir/crl.pem          # The current CRL
private_key        = $dir/private/cakey.pem  # The private key
RANDFILE           = $dir/private/.rand     # private random number file
x509_extensions = usr_cert                 # The extentions to add
                                           # to the cert
```

2. Modify the path values to conform to your custom directory and your custom certificate authority key (private key) and certificate and save your changes:

```
 [ CA_default ]
dir     = .                                # <--CHANGE THIS
certs           = $dir/certs
crl_dir         = $dir/crl
database        = $dir/index.txt
#unique_subject = no
new_certs_dir   = $dir/newcerts
certificate     = $dir/certs/myca.crt      # <--CHANGE THIS
serial          = $dir/serial
#crlnumber      = $dir/crlnumber
crl             = $dir/crl.pem
private_key     = $dir/private/myca.key    # <--CHANGE THIS
RANDFILE        = $dir/private/.rand
x509_extensions = usr_cert
```

18.3.6.4.6 Sign the Certificate Request Now you will sign the certificate request and generate the server's certificate. To do so, proceed as follows:

1. First, copy the `server.csr` to your certificate authority's directory by issuing the following command:

   ```
   # cp server.csr /etc/pki_jungle/myCA/
   ```

2. Change to your certificate authority's directory by issuing the following command:

   ```
   # cd /etc/pki_jungle/myCA/
   ```

3. Then, sign the certificate request by issuing the command:

   ```
   # openssl ca -config openssl.my.cnf -policy policy_anything
   -out certs/server.crt -infiles server.csr
   ```

4. Supply the certificate authority's private key to sign the request. You can check the `openssl.my.cnf` file about what "policy_anything" means. In short, the fields about the `Country`, `State` or `City` are not required to match those of your certificate authority certificate.

   After the steps have been completed, two new files are created:

   - `certs/server.crt.`

     This is the server's certificate, which can be made available publicly.

   - `newcerts/01.pem`

     This is the same certificate, but with the certificate's serial number as a file name. It is not needed.

5. Now, delete the certificate request (`server.csr`) since it is no longer needed.

18.3.6.5 Import the Certificate into Your Keystore

The SSL VPN must import the public key of this server certificate to decrypt the message sent from OAAM.

You must import the Root CA Certificate followed by the certificate which was issued to you by the certificate authority. The name of the root certificate is `myca.crt` and the name of the issued certificate is `server.crt`.

To import a certificate into a keystore, proceed as follows:

1. Change the working directory to *`MW_HOME`*`/jdk160_18/jre/lib/security`.
2. Import the root certificate into your keystore using the following `keytool` command:

   ```
   keytool -importcert -alias rootCA -file myca.crt -keystore cacerts
   -storepass changeit
   ```

   In the preceding syntax:

   - `alias` represents the alias of the Root CA Certificate.
   - `rootCA` -file represents the name of the file that contains the Root CA Certificate.
   - `keystore` represents the name of your keystore.

3. Open `server.crt` in a text editor and remove everything except for content between the `BEGIN CERTIFICATE` and `END CERTIFICATE` tags.

4. Import the issued certificate into your keystore using the following `keytool` command:

```
keytool -importcert -alias OAAMCert -file server.crt -keystore cacerts
-storepass changeit
```

In the preceding syntax:

- `alias` represents the alias of the certificate, which must be the same as the private key alias assigned in Create Private Key for Certificate.
- `server.crt` represents the name of the file that contains the certificate.
- `keystore` represents the name of your keystore.

5. Enter the key password for `<OAAMCert>`.

The certificate reply was installed in keystore.

Note: Ensure that the alias is the same as the one you used when creating the request.

18.3.7 Modify Integration Properties Using the OAAM Administration Console

To define the SAML configuration properties required to establish the integration, proceed as follows:

1. Log in to the OAAM Administration Console.
2. Double-click **Properties** to open the Properties page.
3. Now type `oracle.saml*` in the Name field and click **Search** to search the integration properties.
4. In the search results, click the property you must modify.
5. In the Properties tab, modify the value for the property and click **Save**.

The properties imported as part of integration that need to be modified are shown in Table 18–2, " SAML Integration Properties".

Table 18–2 SAML Integration Properties

Properties	Description
oracle.saml.integration.version	The SAML version used for integration Possible values are `1.1` and `2.0`. The default value is `1.1`. Juniper SA also supports SAML2.0. You must decide the version of SAML to use.
oracle.saml.target.default.url	The target URL (homepage) the user wants to navigate to after successful SAML assertion validation by Juniper SA For example: `https://ag-example-oaam.juniperlabs.local/`
oracle.saml.keystore	The full path of the keystore used for storing the certificate required to sign the assertion. In our case it will be *MW_HOME*`/jdk160_18/jre/lib/security/cacerts`
oracle.saml.keystore.password	The password of the keystore
oracle.saml.keystore.certalias	The alias of the certificate used for assertion

Table 18–2 (Cont.) SAML Integration Properties

Properties	Description
oracle.saml.keystore.privatekeypassword	The private key password
oracle.saml.redirect.post.url	The URL where SAML assertion is posted For example: `https://ag-example-oaam.juniperlabs.local/dana-na/auth/saml-consumer.cgi`
oracle.saml.set.attributes	Indicates if additional attributes need to be sent to the Juniper SA as part of the assertion Possible values are `false` or `true`. The default value is `false`
oracle.saml.user.attributes	List of attributes required to be appended as part of the assertion The property is only used if `oracle.saml.set.attribute` is set to `true`
oracle.saml.attribute.namespace	The name of the namespace used for assertion. The default vale is `JuniperNS`. For SAML1.1 only.
oracle.saml.nameidformat	The `nameid` format used in the SAML assertion The default value is `X509SubjectName`.
oracle.saml.nameidattribute	The `NameID` attribute which identifies the user in the SAML assertion The default value is `distinguishedName`. This must be `distinguishedName` if the `nameid` format is set to `X509SubjectName`.
oracle.saml.issuer.url	The URL of the issuer of SAML This is the machine where the OAAM authentication server is running. For example: `http://abcdefgh.example.com:14300`

18.3.8 Configure Juniper Networks Secure Access (SA)

To configure Juniper Networks Secure Access (SA) for this integration, you must:

- Create SAML 1.1 Authentication Server
- Create a User Realm for SAML
- Create Sign-In Policy

For more information on Juniper SA configuration, see the *Juniper Networks Secure Access Administration Guide* available at

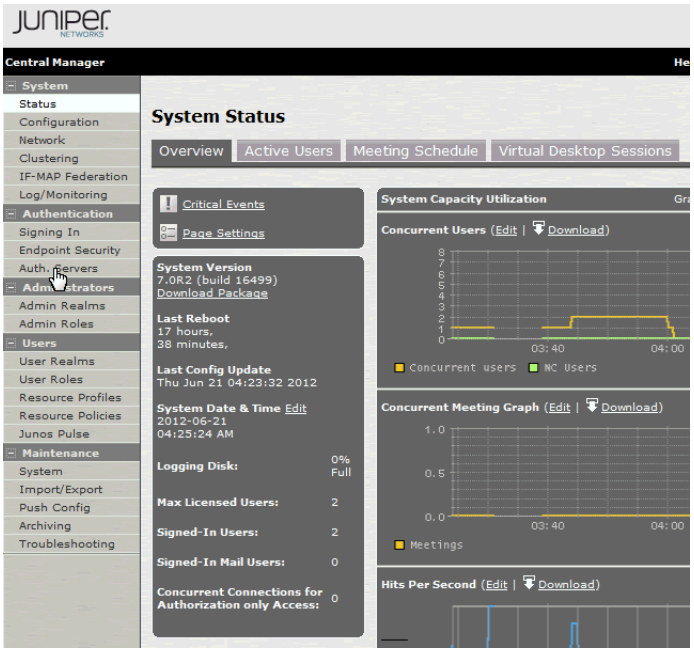
`http://www.juniper.net/techpubs`

18.3.8.1 Create SAML 1.1 Authentication Server

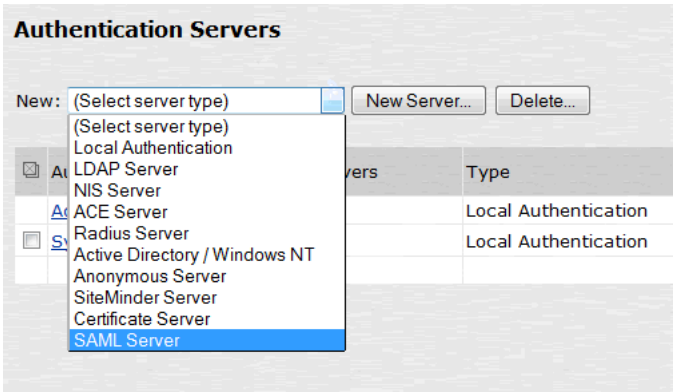
You must create an Authentication Server in Juniper SA. To do so, proceed as follows:

1. Log in to your Juniper SSL VPN Administrator Console.

2. In the Juniper Administration Console in the left pane, expand the **Authentication** menu, and then click **Auth. Servers**.



3. From the **New** drop-down list, select **SAML Server**, and then click **New Server**.



The following dialog is displayed.

4. Define the Authentication Server with the values in Table 18–3.

Table 18–3 Create an Authentication Server

Parameter	Details	Value
Server Name	Name of SAMLServer	OAAM SAML 1.1 Enter same value as shown.
SAML Version	SAML version for authentication server	1.1 Enter same value as shown.
Source Site Inter-Site Transfer Service URL	The entry URL of OAAM server. This is where the user gets redirected to for authentication.	Example: @ `https://ag-exampleoaam.acmegizmo.com:14301/oaam_server/juniperLoginPage.jsp` Specify the host and port according to your environment.
User Name Template	The template used for extracting the value that identifies the authenticated user	`<assertionNameDN.cn>` Enter the same value as shown. You also need to keep '<' and '>' at beginning and end
Allowed Clock Skew (minutes)	Allowed clock skew for assertion.	`30` Enter the same value as shown.
SSO Method	SSO method used for SAML	`Post`
Response Signing Certificate	The certificate used for signing the response.	This is the certificate you obtained from certificate authority. You have imported the same certificate into the Keystore in the step Import the Certificate into Your Keystore.

5. Import the server certificate (for example, `server.crt`) created (in Section 18.3.6.4.6, "Sign the Certificate Request").

6. Click **Save Changes** to save the changes.

18.3.8.2 Create a User Realm for SAML

An authentication realm specifies the conditions that users must meet to sign in. A realm consists of a grouping of authentication resources.

To create a user realm for SAML, proceed as follows:

1. From the Juniper Administration Console, in the left pane, expand the **Users** menu, point to **User Realms**, and then click **New User Realm**.
2. Specify the name as `OAAM SAML 1.1 User Realm`.
3. Select the Authentication Server `OAAM SAML 1.1` that was created in the last step as the authentication server for this user realm.
4. Save the changes.

   Now you should see the newly created user realm.
5. From the Juniper Administration Console, in the left pane, expand the **Users** menu, point to **User Realms**, and then click **OAAM SAML 1.1 User Realm**.
6. From the `OAAM SAML 1.1. User Realm`, click the **Role Mapping** tab to configure one or more role mapping rules.

18.3.8.3 Create Sign-In Policy

Create a Sign-In policy which defines the URL on which you need to go on the Juniper SA to get redirected to OAAM for authentication.

1. To create a sign-in policy, in the Juniper Administration Console, expand the **Authentication** menu, point to **Signing In**, and then click **Sign-in Policies**.
2. Click **New URL** and in the **Sign-in URL** field that is displayed, enter `*/OAAM11/` for the URL.
3. For Sign-in Page, select **Default Sign-in Page**.
4. For Authentication Realm, select the **OAAM SAML 1.1 User Realm** that was created earlier.
5. Click **Save Changes** and make sure it is enabled

18.4 Verify the Integration

Once you have configured all required components, the next step is to test the Login and Forget Password flows. Follow the subsequent steps to verify that OAAM and Juniper SA were integrated successfully.

Test Login Flow

1. Open up a web browser and go to the target/protected resource URL that is protected by Juniper.

   Make sure there are no instances or windows of the Web browser where you logged in to the Juniper Administration Console.

   The target/protected resource URL is the value of the `oracle.saml.target.default.url` property specified in the OAAM Administration Console.

   The user will be taken to the OAAM Server login page.

2. Complete the login process.

   It should take you to the Juniper Sign In page that displays the username and shows the sign-in page that has bookmarks and link to resources.

Test Forgot Password Flow

1. Open a Web browser and go to the target/protected resource URL that is protected by Juniper.

   Make sure there are no instances or windows of the Web browser where you logged in to the Juniper Administration Console.

   The target/protected resource URL is the value of the `oracle.saml.target.default.url` property specified in OAAM Administration Console.

   The user will be taken to the OAAM Server login page.

2. Enter the username and click **Continue**.
3. In the Password page click the **Forgot your password** link.
4. Answer the challenge questions or OTP.

   After successfully answering the challenge, the user will be allowed to change the password and log in to the Juniper Sign In page.

18.5 Debug the Integration

To debug the integration on the OAAM end, enable the debug logs by following these steps:

1. Log in to Oracle Enterprise Manager Fusion Middleware Control. For example:

   ```
   http://host.domain.com:7001/em/
   ```

   For information on using Fusion Middleware Control, see *Oracle Fusion Middleware Administrator's Guide*.

2. Expand the **WebLogic Domain** node and click the **OAAM Server**.
3. Right-click to select **Log** and then **Log Configuration** to open the log configuration for the OAAM Server.

The default logging will be set to `FINER` level

4. Change the log level to `NOTIFICATION:1(INFO)`. Also, select **Persist log level state across component restarts**. Click the **Apply** button to save changes.

The debug logs will be in located in:

MW_HOME`/user_projects/domains/`*YOURDOMAIN*`/servers/oaam_server/logs/oaam_server-diagnostic.log`

18.6 Troubleshooting Common Problems

This section describes common problems you might encounter in an Oracle Adaptive Access Manager and Juniper Networks Secure Access (SA) integrated environment and explains how to solve them.

In addition to this section, review the *Oracle Fusion Middleware Error Messages Reference* for information about the error messages you may encounter.

For information about additional troubleshooting resources, see Section 29.1, "Using My Oracle Support for Additional Troubleshooting Information."

18.6.1 Juniper SA and OAAM Clock Synchronization

Ensure that the Juniper SA and OAAM Servers system clocks are synchronized. Juniper SA system clock should not be ahead of OAAM Server clock. Refer to the *Juniper Networks Secure Access Administration Guide* to reset the date and time in the Juniper application. You can visit the Juniper Technical Documentation website at

`http://www.juniper.net/techpubs`

18.6.2 Absence of a Correct Certificate on Juniper

After the OAAM flow has completed, the user is not redirected to the protected resource and an `InvalidCryptoException` is seen in the Web browser:

The logs seen in the Juniper Administration Console is as follows:

```
Logs(Juniper admin->Log/monitoring->Events): ERR24377 : Caught a SAML exception
'InvalidCryptoException' while verifying response. Error:
SAMLSignedObject::verify() failed
to validate signature value.
```

Cause

The correct certificate is absent on Juniper.

Solution

Ensure that server certificate (for example, `server.crt`) has been uploaded in Juniper. Refer to Section 18.3.8.1, "Create SAML 1.1 Authentication Server".

18.6.3 Signing Failure in SAML Response

After entering the password in OAAM, a screen with the following message appears:

```
There has been an error reaching your destination
```

The following exception is seen in the OAAM server log file (namely `oaam_server_server1.log`):

```
java.lang.NullPointerException
```

```
at java.io.PrintWriter.write(PrintWriter.java:429)
at jsp_servlet.__samlsubmit._jspService(__samlsubmit.java:96)
at weblogic.servlet.jsp.JspBase.service(JspBase.java:34)
at weblogic.servlet.internal.StubSecurityHelper$ServletServiceAction.run(StubSecurityHelper.java:227)
at weblogic.servlet.internal.StubSecurityHelper.invokeServlet(StubSecurityHelper.java:125)
at weblogic.servlet.internal.ServletStubImpl.execute(ServletStubImpl.java:300)
at weblogic.servlet.internal.ServletStubImpl.onAddToMapException(ServletStubImpl.java:416)
at weblogic.servlet.internal.ServletStubImpl.execute(ServletStubImpl.java:326)
at weblogic.servlet.internal.ServletStubImpl.execute(ServletStubImpl.java:183)
```

Cause

There was a signing failure in the SAML response.

Solution

Ensure that the following properties are set to the right values:

For information, refer to Table 18–2, " SAML Integration Properties".

Properties	Description
oracle.saml.keystore	The full path of the keystore used for storing the certificate required to sign the assertion. In our case it will be *MW_HOME*/jdk160_18/jre/lib/security/cacerts
oracle.saml.keystore.password	The password of the keystore
oracle.saml.keystore.certalias	The alias of the certificate used for assertion
oracle.saml.keystore.privatekeypassword	The private key password

18.6.4 Entry Point URL for OAAM

On accessing the protected resource, the following error message is seen:

```
There has been an error reaching destination
```

Cause

The entry point URL for OAAM was not set correctly in Juniper.

Solution

Check the URL in the browser. If it is not set to `https://`*OAAM_HOST*`:`*OAAM_PORT*`/oaam_server/juniperLoginPage.jsp`, it must be changed to this correct value.

To do this:

1. Login to Juniper Administration Console.
2. Click **Auth servers** on the left pane.
3. Select the server corresponding to OAAM. For example, **OAAM server**.
4. Set the **Source Site Inter-Site Transfer Service URL** to

   `https://`*OAAM_HOST*`:`*OAAM_PORT*`/oaam_server/juniperLoginPage.jsp`

5. Save the changes.

19

Java Message Service Queue (JMSQ) Integration

Customers with access monitoring requirements involving multiple applications and data sources now have the ability to take a proactive security and compliance posture. Using the provided Java Message Service Queue (JMSQ) customers can implement near real-time risk analysis to actively identify suspected fraud or misuse.

This chapter describes how to integrate OAAM and Java Message Service Queue (JMSQ) for asynchronous integration. It contains the following sections:

19.1 JMS Definitions

Table 19–1 lists JMS terms and definitions. For in-depth information about Java Message Service, see *Oracle Fusion Middleware Configuring and Managing JMS for Oracle WebLogic Server*.

Table 19–1 JMS Definitions

Term	Definition
Messaging	Messaging is a method of communication between software components or applications. A messaging system is a peer-to-peer facility: A messaging client can send messages to, and receive messages from, any other client. Each client connects to a messaging agent that provides facilities for creating, sending, receiving, and reading messages. Messaging enables distributed communication that is loosely coupled. A component sends a message to a destination, and the recipient can retrieve the message from the destination. However, the sender and the receiver do not have to be available at the same time in order to communicate. In fact, the sender does not need to know anything about the receiver; nor does the receiver need to know anything about the sender. The sender and the receiver need to know only what message format and what destination to use. Messaging also differs from electronic mail (email), which is a method of communication between people or between software applications and people. Messaging is used for communication between software applications or software components.
Java Message Service (JMS)	Java Message Service (JMS) is a Java API that allows applications to create, send, receive, and read messages using reliable, asynchronous, loosely coupled communication. The JMS API defines a common set of interfaces and associated semantics that allow programs written in the Java programming language to communicate with other messaging implementations.
JMS Queues	JMS Queues are message queues that allow software or applications to exchange information asynchronously. Each message is addressed to a specific queue, and receiving clients extract messages from the queues established to hold their messages. Queues retain all messages sent to them until the messages are consumed or expire.
JNDI	JNDI is a java naming and directory service. The JMS queue can be accessed using the JNDI names.
JMS Queue Destination	Applications send messages to the queue. Provider stores one copy of each message until OAAM receives the message.
JMS Listener	The JMS listener is configured to listen to JMS queues for messages in XML format. A client can register a message listener with a consumer. A message listener is similar to an event listener. Whenever a message arrives at the destination, the JMS provider delivers the message; then, the contents of the message are acted upon. The XML schema in Section 19.4.2, "XML Schema Example for Message Formats" provides details about the message format.
OAAM JMS Message	OAAM JMS message contents look similar to the Oracle Adaptive Access Manager Web Services API calls. The XML schema in Section 19.4.2, "XML Schema Example for Message Formats" provides details about the message format.

19.2 Install the Asynchronous Integration Option

Table 19–2 lists a summary of the tasks for installing the Asynchronous Integration Option.

Table 19–2 Asynchronous Integration Option Installation

No.	Task	Information
1	Review prerequisites.	For information, refer to Pre-requisites.
2	Install the Asynchronous Integration Option.	For information, refer to Installing the Asynchronous Integration Option.
3	Set Up the JMS Queues.	For information, refer to Setting Up JMS Queues.
4	Update the OAAM Extensions Library.	For information, refer to Updating the OAAM Extensions Library.
5	Update the database.	For information, refer to Updating the OAAM Database.

19.2.1 Pre-requisites

Ensure that Oracle Adaptive Access Manager 11*g* is installed and configured before proceeding. This section contains the steps required to install the Asynchronous Integration Option.

The Asynchronous Integration Option includes various reports as Oracle Business Intelligence Publisher report templates. Ensure that Oracle Business Intelligence Publisher is installed and configured before proceeding with installation of the Asynchronous Integration Option. For information on installing Oracle Business Intelligence Publisher, see *Oracle Fusion Middleware Installation Guide for Oracle Business Intelligence*.

19.2.2 Installing the Asynchronous Integration Option

The Asynchronous Integration Option contains the `osg_integration_kit.zip` file.

To install the Asynchronous Integration Option, proceed as follows:

1. Create a work directory called `osg_install` on the machine where OAAM is installed. The directory can be created anywhere if it is outside the installation folder.
2. Create an `osg_integration_kit` directory inside the `osg_install` directory.
3. Locate `osg_integration_kit.zip`, which is located in the *`IAM_Home`*`/oaam/oaam_extensions/generic` directory.
4. Extract the contents of `osg_integration_kit.zip` to the `osg_install/osg_integration_kit` directory.

19.2.3 Updating the OAAM Extensions Library

The asynchronous execution functionality is implemented as an Oracle Adaptive Access Manager extension. Follow the subsequent steps to replace the default extension library:

1. Shut down all Oracle Adaptive Access Manager servers. For example, `oaam_server_server1` and `oaam_admin_server1`.
2. Start the WebLogic Server where Oracle Adaptive Access Manager is deployed and log in to the Oracle WebLogic Administration Console.

3. Click **Lock & Edit**.
4. Select `oracle.oaam.extensions` and click **Delete**.
5. Navigate to **Domain Environment** > **Deployments** and lock the console.
6. Click the **Install** button.
7. Browse to the location of the `osg_oaam_extensions.war` file and select it by clicking the option next to the WAR file and clicking **Next**.
8. Ensure **Install this deployment as a library** is selected and click **Next**.
9. Select all OAAM servers as deployment targets. For example, `oaam_admin_server1` and `oaam_server_server1`.
10. Click **Next** again to accept the defaults in this next page and then click **Finish**.
11. Click the **Save** button and then A**ctivate Changes**.
12. Start all necessary Oracle Adaptive Access Manager servers.

19.2.4 Setting Up JMS Queues

OAAM uses JMS (Java Message Service) queues as one of the integration mechanisms. OAAM listens on one or more JMS queues for XML messages. For example, an electronic patient medical records (EMR) will need a message queue to transmit transaction details that will be used by the OAAM server. For information on the XML schema and listener configuration, see Section 19.3, "JMS Integration."

With the default configuration included in `osg_oaam_extension.war`, OAAM listens for messages in a JMS queue with JNDI name `jms/oaamDefaultQueue` at `t3://localhost:7001`. Review this configuration and update as necessary for your deployment, per the details given in Section 19.3, "JMS Integration."

Ensure that the JMS queues specified in the listeners configuration exists and are active. If any do not exist, create them in the appropriate WebLogic Server. For information on setting up a JMS Queue on a WebLogic Server, see *Oracle Fusion Middleware Configuring and Managing JMS for Oracle WebLogic Server*.

19.2.5 Updating the OAAM Database

Database views for the entities and transactions can be created for use in rule conditions and reports. For information on these database views, see Section 19.5, "Database Views for Entities and Transactions." Database privileges of the OAAM database user must be updated for view creation and other database operations.

To grant the necessary privileges, log in to Oracle database with administrator credentials and run the following SQL statement, replacing `dev_oaam` with the OAAM database schema:

```
grant create view to dev_oaam
```

19.3 JMS Integration

With the JMS listener implementation in an asynchronous deployment, you can configure Oracle Adaptive Access Manager to listen to JMS queues (or topics) for messages in XML format. The XML schema shown in Section 19.4.2, "XML Schema Example for Message Formats" provides details on the message format. For details on the parameters, you can refer to *Oracle Fusion Middleware Java API Reference for Oracle Adaptive Access Manager*. JMS message contents is similar to OAAM Web Services API calls.

19.3.1 Web Services API

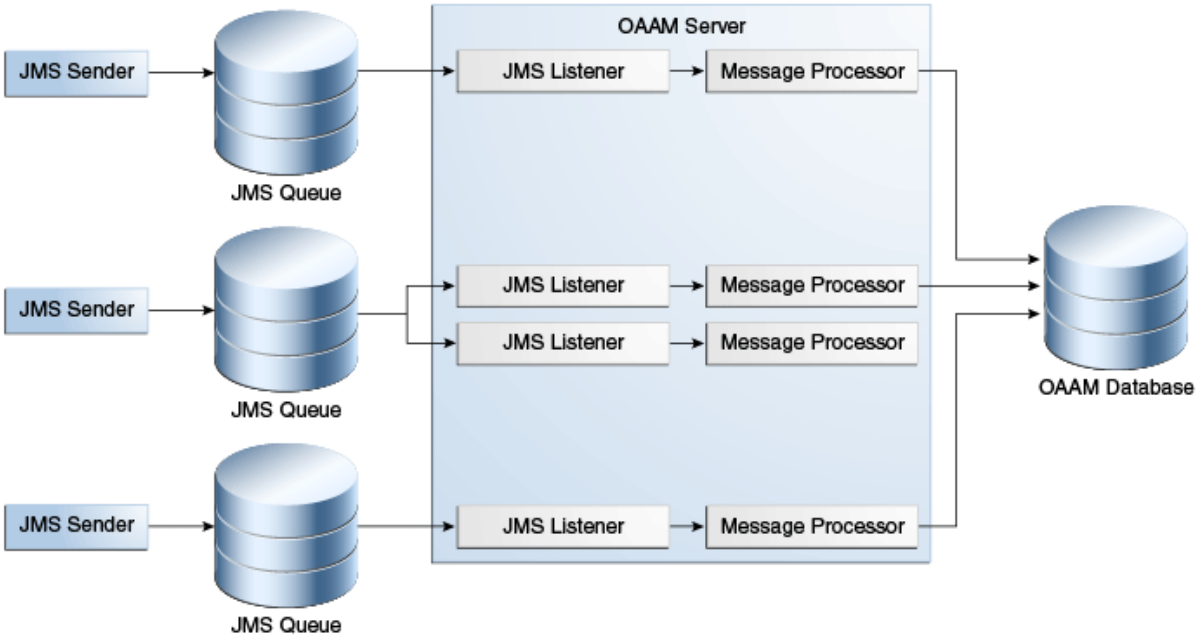
The following is a subset of Web Services APIs available through JMS:

- VCryptTracker.updateLog
- VCryptTracker.updateEntity
- VCryptTracker.createTransaction
- VCryptRulesEngine.processRules

19.3.2 JMS Integration Diagram

The subsequent diagram provides an overview of the JMS integration with OAAM.

Figure 19–1 JMS integration with OAAM



The flow of interaction is as follows:

1. The application (JMS Sender) sends a message to the JMS Queue. It identifies the queue destination by the JNDI namespace.
2. The queuing system receives the message from the JMS Sender and routes the message to the destination.
3. OAAM Server listens for the message with the configured JMS Listener.
4. The message is processed by the Message Processor.
5. Information is loaded into the database as transaction or login data.
6. Rules are then run on the login and transaction data offline.

19.3.3 Registering the JMS Listener

You can configure various aspects of JMS integration using Oracle Adaptive Access Manager properties and user-defined enums. For information on user-defined enums, see Section 7.1, "Customizing or Extending OAAM By Editing Enums."

Table 19–3 shows the list of JMS configuration properties. For each queue (or topic) to be monitored, one listener must be configured by adding an enum element in the user-defined enum `oracle.oaam.jms.listeners.enum`. Any changes to the listener list or properties require the OAAM Server where the listeners run to be restarted.

Table 19–3 JMS Configuration Properties

Property Name	Description
jms.message.processor.default.user	When the `loginId` field is not specified, in the `VCryptTracker.updateLog` message, the value in this property will be used. Default value: `JmsDefaultUser`
jms.message.processor.default.usergroup	When the `groupId` field is not specified in the `VCryptTracker.updateLog` message, the value in this property will be used. Default value: `default`
jms.message.processor.default.ip	When the `remoteIPAddr` field is not specified in the `VCryptTracker.updateLog` message, the value in this property will be used. Default value: `127.0.0.1`
jms.message.processor.default.clientver	When the `clientVersion` field is not specified in the `VCryptTracker.updateLog` message, the value in this property will be used. Default value: 1.0
jms.message.processor.default.authtype	When the `clientType` field is not specified in the `VCryptTracker.updateLog` message, the value in this property will be used. Default value: normal
jms.message.processor.default.sessionid	When the `requestId` field is not specified in the `VCryptTracker.updateLog` message, the value in this property will be used. If no value is specified, a value generated using the `remoteIPAddr` field value will be used. In the case of multiple message being sent in a `MessageList`, when the `requestId` field is not specified for `createTransaction/updateEntity/processRules` messages, the `requestId` used in the earlier `updateLog` message, if available, will be used.
jms.message.processor.default.browser.fingerprint	When the `fingerPrint` field is not specified in the `VCryptTracker.updateLog` message, the value in this property will be used.
jms.message.processor.default.flash.fingerprint	When the `fingerPrint2` field is not specified in the `VCryptTracker.updateLog` message, the value in this property will be used.
oracle.oaam.jms.listeners.default.initial.context.factory	Name of the Java class that implements the initial context factory. This value will be used to initialize the Java Naming context. Default value: `weblogic.jndi.WLInitialContextFactory`
oracle.oaam.jms.listeners.default.connection.factory	JNDI name of the JMS connection factory used to create queue/topic connections. Default value: `weblogic.jms.ConnectionFactory`

Table 19–3 (Cont.) JMS Configuration Properties

Property Name	Description
oracle.oaam.jms.listeners.enum.lsnr_1	Defines a new listener named `lsnr_1`. Various attributes for this listener can be specified using the properties listed. Valid Value: an integer which is not assigned to any other element in this enum
oracle.oaam.jms.listeners.enum. lsnr_1.type	Specify whether the listener will be connecting to a JMS queue or a JMS topic. Valid values: `queue` or `topic`
oracle.oaam.jms.listeners.enum. lsnr_1.url	JNDI provider URL to resolve the queue (or topic) names.
oracle.oaam.jms.listeners.enum.lsnr_1.jndiname	JNDI name of the queue or topic
oracle.oaam.jms.listeners.enum.lsnr_ 1.initial.context.factory	Name of the Java class that implements the initial context factory. This value will be used to initialize Java Naming context. Default value: value of property `oracle.oaam.jms.listeners.default.initial.context.factory`
oracle.oaam.jms.listeners.enum.lsnr_ 1.connection.factory	JNDI name of the JMS connection factory used to create queue/topic connections. Default value: value of property `oracle.oaam.jms.listeners.default.connection.factory`
oracle.oaam.jms.listeners.enum.lsnr_1.processor	Name of the Java class that implements the message processor interface. An instance of this class will be created to process messages received by this listener. Default value: `oracle.oaam.jms.JmsDefaultMessageProcessor`
oracle.oaam.jms.listeners.enum.lsnr_1.instancecount	Number of listeners to create to process messages from the specified queue/topic. Default value: `1`

19.3.4 Configuring Message Processor

Configure Message Processor properties using Oracle Adaptive Access Manager properties and user-defined enums. For information on user-defined enums, see Section 7.1, "Customizing or Extending OAAM By Editing Enums."

The OAAM default JMS message processor processes only the messages of type `javax.jms.TextMessage`. Other types of messages are ignored by the JMS message processor.

To process other type of messages, you must implement a custom processor by extending either `oracle.oaam.jms.JmsAbstractMessageProcessor` or `oracle.oaam.jms.JmsDefaultMessageProcessor` and associating the processor with a `JmsListener`.

In addition, the default JMS message processor processes only if the contexts of the `TextMessage` is a XML string that conforms to the XML schema as shown in Section 19.4.2, "XML Schema Example for Message Formats."

19.4 JMS Messages

This section includes the following topics:

- JMS Message Examples
- XML Schema Example for Message Formats
- Sending a Message to a JMS Queue

19.4.1 JMS Message Examples

This section provides JMS message examples. It includes the following topics:

- VCryptTracker.updateLog
- VCryptTracker.updateEntity
- VCryptTracker.createTransaction
- VCryptRulesEngine.processRules
- MessageList

These message examples may not include all data elements supported for the messages. For a complete list of supported data elements, see Section 19.4.2, "XML Schema Example for Message Formats."

19.4.1.1 VCryptTracker.updateLog

The `VCryptTracker.updateLog` message is used to create or update a user-session (login) in Oracle Adaptive Access Manager. If no value is specified for the `requestId` data element, a unique value will be generated and used.

A `VCryptTracker.updateLog` message example is shown as follows:

```
<?xml version="1.0"?>
<OaamJmsMessage>
  <VCryptTracker.updateLog>
    <UpdateAuthResultRequest>
      <requestId>20110721_00_9004_terminal_1</requestId>
      <requestTime>07/21/2011 00:21:01</requestTime>
      <userId>9004</userId>
      <loginId>jjames</loginId>
      <isSecure>false</isSecure>
      <groupId>HealthcareUsers</groupId>
      <result>0</result>
      <clientType>10</clientType>
      <clientVersion>1.0</clientVersion>
      <remoteIPAddr>192.168.0.0</remoteIPAddr>
      <remoteHost>server.domain.com</remoteHost>
    </UpdateAuthResultRequest>
  </VCryptTracker.updateLog>
<OaamJmsMessage>
```

This message is the JMS equivalent of the OAAM API `VCryptTracker.updateLog()`. For information on the data elements (parameters), see *Oracle Fusion Middleware Java API Reference for Oracle Adaptive Access Manager*.

19.4.1.2 VCryptTracker.updateEntity

The `VCryptTracker.updateEntity` message is used to create or update a user-defined entity instance in Oracle Adaptive Access Manager.

This is a `VCryptTracker.updateEntity` message example:

```
<?xml version="1.0"?>
<OaamJmsMessage>
```

```
  <VCryptTracker.updateEntity>
    <updateEntity>
      <entityDefKey>Patient</entityDefKey>
      <contexts>
        <context>
          <name>Patient_ID</name>
          <value>21600481</value>
        </context>
        <context>
          <name>MR_Number</name>
          <value>21600481</value>
        </context>
        <context>
          <name>Short_Name</name>
          <value>Jane</value>
        </context>
        <context>
          <name>Last_Name</name>
          <value>Celebrity</value>
        </context>
        <context>
          <name>First_Name</name>
          <value>Jane</value>
        </context>
        <context>
          <name>Phone_Number</name>
          <value>603.555.0100</value>
        </context>
        <context>
          <name>Email_Address</name>
          <value>Jane.Celebrity@hotmail.com</value>
        </context>
        <context>
          <name>Date_Of_Birth</name>
          <value>1979-05-12 00:00:00 -0800</value>
        </context>
        <context>
          <name>Confidential_Indicator</name>
          <value>true</value>
        </context>
        <context>
          <name>homeAddr.Line1</name>
          <value>6819 Park Blvd</value>
        </context>
        <context>
          <name>homeAddr.City</name>
          <value>Los Angeles</value>
        </context>
        <context>
          <name>homeAddr.State</name>
          <value>California</value>
        </context>
        <context>
          <name>homeAddr.Zip</name>
          <value>90001</value>
        </context>
        <context>
    </updateEntity>
  </VCryptTracker.updateEntity>
</OaamJmsMessage>
```

This message is the JMS equivalent for the OAAM API `VCryptTracker.updateEntity()`. For information on the data elements (parameters), see *Oracle Fusion Middleware Java API Reference for Oracle Adaptive Access Manager*.

19.4.1.3 VCryptTracker.createTransaction

The `VCryptTracker.createTransaction` message is used to create a transaction in Oracle Adaptive Access Manager.

This is a `VCryptTracker.createTransaction` message example:

```
<?xml version="1.0"?>
<OaamJmsMessage>
  <VCryptTracker.createTransaction>
    <createTransaction>
      <requestId>20110721_00_9004_terminal_1</requestId>
      <requestTime>07/21/2011 00:21:01</requestTime>
      <transactionDefKey>pat_rec_acc</transactionDefKey>
      <status>0</status>
      <contexts>
        <context>
          <name>Person_ID</name>
          <value>9004</value>
        </context>
        <context>
          <name>Patient_ID</name>
          <value>21600481</value>
        </context>
        <context>
          <name>Action</name>
          <value>View_Records</value>
        </context>
        <context>
          <name>Application_ID</name>
          <value>Healthcare_App</value>
        </context>
        <context>
          <name>Terminal_ID</name>
          <value>terminal_1</value>
        </context>
        <context>
          <name>Item_Key</name>
          <value>image-x-20110720-156</value>
        </context>
        <context>
          <name>Is_Restricted_Item</name>
          <value>false</value>
        </context>
        <contexts>
    </createTransaction>
  </VCryptTracker.createTransaction>
</OaamJmsMessage>
```

This message is the JMS equivalent for the OAAM API `VCryptTracker.createTransaction()`. For information on the data elements (parameters), see *Oracle Fusion Middleware Java API Reference for Oracle Adaptive Access Manager*.

19.4.1.4 VCryptRulesEngine.processRules

The `VCryptRulesEngine.processRules` message is used to run OAAM rules.

This is a `VCryptRulesEngine.processRules` message example:

```
<?xml version="1.0"?>
<OaamJmsMessage>
  <VCryptRulesEngine.processRules>
    <ProcessRulesRequest>
      <requestId>20110721_00_9004_terminal_1</requestId>
      <requestTime>07/21/2011 00:21:01</requestTime>
      <profileTypeList>
        <profileType>800</profileType>
      </profileTypeList>
      <contexts></contexts>
    </ProcessRulesRequest>
  </VCryptRulesEngine.processRules>
</OaamJmsMessage>
```

This message is the JMS equivalent for the OAAM API `VCryptRulesEngine.processRules()`. For information on the data elements (parameters), see *Oracle Fusion Middleware Java API Reference for Oracle Adaptive Access Manager*.

19.4.1.5 MessageList

The `MessageList` message is used to send one or more messages in one JMS message.

This is a `MessageList` message example:

```
<?xml version="1.0"?>
<OaamJmsMessage>
  <MessageList>
    <VCryptTracker.updateLog>
    </VCryptTracker.updateLog>

    <VCryptTracker.updateEntity>
    </VCryptTracker.updateEntity>

    <VCryptTracker.updateEntity>
    </VCryptTracker.updateEntity>

    <VCryptTracker.createTransaction>
    </VCryptTracker.createTransaction>

    <VCryptRulesEngine.processRules>
    </VCryptRulesEngine.processRules>
  </MessageList>
</OaamJmsMessage>
```

This message is the JMS equivalent of calling multiple OAAM API calls in a batch.

In addition to batching the messages, the `MessageList` message also provides useful default values for `requestId` and `transactionLogId` data elements.

- When `transactionLogId` is not explicitly specified in the `processRules` message, the value returned from the earlier `createTranasction` message, if available, is used.
- When `requestId` is not explicitly specified in the `createTransaction`, `processRules` or `updateEntity` message, the value of the `requestId` data element from the previous message (within this `MessageList`) is used

19.4.2 XML Schema Example for Message Formats

The following XML schema example shows the details on the message format.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema id="OaamJmsMessage"
           xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="OaamJmsMessage" type="MessageList"/>

    <xs:complexType name="MessageList">
      <xs:sequence>
          <xs:element name="MessageList" type="MessageList"
                    minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="VCryptTracker.updateLog"
                    type="updateLog"
                    minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="VCryptTracker.createTransaction"
                    type="createTransaction"
                    minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="VCryptRulesEngine.processRules"
                    type="processRules"
                    minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="VCryptTracker.updateEntity"
                    type="updateEntity"
                    minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
  </xs:complexType>

  <xs:complexType name="updateLog">
    <xs:sequence>
    <xs:element name="UpdateAuthResultRequest">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="requestId" type="xs:string"
                      minOccurs="0" maxOccurs="1"/>
          <xs:element name="requestTime" type="xs:string"
                      minOccurs="1" maxOccurs="1"/>
          <xs:element name="userId" type="xs:string"
                      minOccurs="0" maxOccurs="1"/>
          <xs:element name="loginId" type="xs:string"
                      minOccurs="0" maxOccurs="1"/>
          <xs:element name="isSecure" type="xs:boolean"
                      minOccurs="0" maxOccurs="1"/>
          <xs:element name="groupId" type="xs:string"
                      minOccurs="0" maxOccurs="1"/>
          <xs:element name="result" type="xs:integer"
                      minOccurs="0" maxOccurs="1"/>
          <xs:element name="secureCookie" type="xs:string"
                      minOccurs="0" maxOccurs="1"/>
          <xs:element name="digitalCookie" type="xs:string"
                      minOccurs="0" maxOccurs="1"/>
          <xs:element name="clientType" type="xs:integer"
                      minOccurs="0" maxOccurs="1"/>
          <xs:element name="clientVersion" type="xs:string"
                      minOccurs="0" maxOccurs="1"/>
          <xs:element name="remoteIPAddr" type="xs:string"
                      minOccurs="0" maxOccurs="1"/>
          <xs:element name="remoteHost" type="xs:string"
                      minOccurs="0" maxOccurs="1"/>
          <xs:element name="fingerPrintType" type="xs:integer"
                      minOccurs="0" maxOccurs="1"/>
```

```
        <xs:element name="fingerPrint" type="xs:string"
                    minOccurs="0" maxOccurs="1"/>
        <xs:element name="fingerPrintType2" type="xs:integer"
                    minOccurs="0" maxOccurs="1"/>
        <xs:element name="fingerPrint2" type="xs:string"
                    minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="createTransaction">
  <xs:sequence>
  <xs:element name="createTransaction">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="requestId" type="xs:string"
                    minOccurs="0" maxOccurs="1"/>
        <xs:element name="requestTime" type="xs:string"
                    minOccurs="0" maxOccurs="1"/>
        <xs:element name="transactionDefKey" type="xs:string"
                    minOccurs="1" maxOccurs="1"/>
        <xs:element name="status" type="xs:integer"
                    minOccurs="0" maxOccurs="1"/>
        <xs:element name="contexts"
                    minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="context"
                          minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="name" type="xs:string"
                          minOccurs="1" maxOccurs="1"/>
                    <xs:element name="value" type="xs:string"
                          minOccurs="1" maxOccurs="1"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="updateEntity">
  <xs:sequence>
   <xs:element name="updateEntity">
     <xs:complexType>
      <xs:sequence>
        <xs:element name="requestId" type="xs:string"
                    minOccurs="0" maxOccurs="1"/>
        <xs:element name="entityDefKey" type="xs:string"
                    minOccurs="1" maxOccurs="1"/>
        <xs:element name="status" type="xs:integer"
                    minOccurs="0" maxOccurs="1"/>
```

```
            <xs:element name="isReplace" type="xs:boolean"
                        minOccurs="0" maxOccurs="1"/>
            <xs:element name="createTime" type="xs:string"
                        minOccurs="0" maxOccurs="1"/>
            <xs:element name="updateTime" type="xs:string"
                        minOccurs="0" maxOccurs="1"/>
            <xs:element name="unlinkEntities"
                        minOccurs="0" maxOccurs="1">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="entityRelation"
                                type="xs:string"
                                minOccurs="0"
                                maxOccurs="unbounded"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="contexts"
                        minOccurs="1" maxOccurs="1">
                <xs:complexType>
                <xs:sequence>
                    <xs:element name="context"
                                minOccurs="0 maxOccurs="unbounded">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="name" type="xs:string"
                                      minOccurs="1" maxOccurs="1"/>
                          <xs:element name="value" type="xs:string"
                                      minOccurs="1" maxOccurs="1"/>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>

    <xs:complexType name="processRules">
      <xs:sequence>
      <xs:element name="ProcessRulesRequest">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="requestId" type="xs:string"
                        minOccurs="0" maxOccurs="1"/>
            <xs:element name="requestTime" type="xs:string"
                        minOccurs="0" maxOccurs="1"/>
            <xs:element name="transactionLogId" type="xs:integer"
                        minOccurs="0" maxOccurs="1"/>
            <xs:element name="extTransactionId" type="xs:string"
                        minOccurs="0" maxOccurs="1"/>
            <xs:element name="profileTypeList"
                        minOccurs="1" maxOccurs="1">
              <xs:complexType>
                <xs:sequence>
                <xs:element name="profileType" type="xs:integer"
                            minOccurs="1" maxOccurs="unbounded"/>
```

```
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  </xs:sequence>
 </xs:complexType>

</xs:schema>
```

19.4.3 Sending a Message to a JMS Queue

The following java code is an example of how to write a message to send to the JMS queue.

```
/* XML Message */
String xmlString = ;

/* get reference to queue from its JNDI name */
javax.jms.Queue queue = ;

/* get a queue connection from connection factory */
QueueConnection queueConn = ;

QueueSession queueSess   = conn.createQueueSession(...);
QueueSender  queueSender = queueSess.createSender(queue);

TextMessage msg = queueSess.createTextMessage(xmlString);

queueSender.send(msg);
```

19.5 Database Views for Entities and Transactions

Users can define entities and transactions in Oracle Adaptive Access Manager with any number of data fields. In addition, transactions can also be defined to reference entities. Oracle Adaptive Access Manager persists the entity and transaction data in the database. The OAAM database schema is designed to store any type of entity and transaction data. However this generic schema makes it challenging to write SQL queries to work with the entity and transaction data.

Oracle Adaptive Access Manager provides a command line tool to generate the SQL script file which contains SQL queries to create views for entities and transactions in Oracle Adaptive Access Manager. These database views makes is easier to query the transaction and entity data and create reports using Oracle Business Intelligence Publisher.

19.5.1 Generating SQL Script File

To generate the SQL script:

1. Set up the OAAM CLI environment.

   For instructions on setting up the OAAM command line environment, see "Setting Up the CLI Environment" in *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*.

2. Generate the SQL script file.

To generate the SQL script, run the following command from the OAAM CLI working folder:

```
$ sh generateTrxEntityViewsSQL.sh
```

The default file name is `createTrxEntityViews.sql`. Optionally, the administrator can change the default filename by adding property with the name `oaam.trxentityview.filename` with required filename as the value.

3. Create the database views for entities and transactions.

   Follow these steps to create the database views for entities and transactions stored in the OAAM database:

   a. Ensure that the OAAM database schema has privileges to create views.

   b. Connect to database using the OAAM database schema user.

      For example:

      ```
      sqlplus DEV_OAAM/PASSWORD
      ```

   c. Run the `createTrxEntityViews.sql` script:

      ```
      SQL>@ createTrxEntityViews.sql
      ```

      This script creates database views for each transaction and entity defined in the OAAM database.

19.5.2 Entity View Details

For each entity defined in Oracle Adaptive Access Manager, one view will be created with the name oaam_ent_*entity_key*. *entity_key* will be replaced by the key of the entity as defined in Oracle Adaptive Access Manager.

The created view will contain one column for each data defined in the entity. For the naming convention of the data columns and the view name, see Section 19.5.4, "Identifiers."

In addition to the data columns, the created view will contain the following columns:

- entity_id: unique identifier for the entity instance
- create_time: time the entity was created
- update_time: time of last update to the entity

19.5.3 Transaction View Details

For each transaction defined in Oracle Adaptive Access Manager, one view will be created with the name oaam_trx_*transaction_key*. *transaction_key* will be replaced by the name of the transaction as defined in Oracle Adaptive Access Manager.

The created view will contain one column for each data defined in the transaction. For the naming convention of the data columns and the view name, seeSection 19.5.4, "Identifiers."

The created view will contain one column for each entity referenced in the transaction to store the `entity_id` of the referenced entity, that is, the `entity_id` column in the `oaam_ent_entity_key` view. Spaces in the instance names will be replaced with an underscore in the column name.

In addition to the data columns, the created view will contain the following columns:

- log_id: unique identifier for the transaction
- user_id: user who performed this transaction
- request_id: session in which this transaction was performed
- ext_trx_id: external ID of this transaction
- status: status of the transaction
- create_time: time the entity was created
- update_time: time of last update to the entity
- created_hour: create time truncated to nearest hour
- created_day: create time truncated to nearest day
- created_week: create time truncated to nearest week
- created_month: create time truncated to nearest month
- created_year: create time truncated to nearest year

19.5.4 Identifiers

The Oracle database limits the length of identifiers (table, view, and column names) to 30 characters. To ensure that the views created by this script comply with this requirement, you should limit the length of entity, transaction, and data field names to the following:

- entity: 21 (view names will be `oaam_ent_` + *`entity_key`*)
- transaction: 21 (view names will be `oaam_trx_` + *`transaction_key`*)
- data-field: 28 (column name will be `d_` + *`data_element_name`*)
- entity-ref: 20 (column name will be *`relationship_name`* + `_entity_id`)

If the script finds any names longer than these limits, the script will trim the identifier. Look for such trimmed column and view names while writing SQL queries on the created views.

Space, dash ("-") and period (".") characters in the names will be replaced with an underscore.

19.6 Python Rule Condition

You can use the Python rule condition to evaluate the python expression using OAAM objects.

19.6.1 Python Expression

The Asynchronous Integration Option includes an OAAM condition to execute a Python expression. You must import the condition from the `osg_install/osg_integration_kit/osg_rule_conditions.zip` file. Python expressions enable the writing of sophisticated conditions without having to write custom Java code. Expressions used in this condition can contain any Python construct, including function calls, SQL queries, multiple statements, and so on. The only requirement is that the expression must return the condition result (a boolean value) in a variable named *`oaamResult`*. Expressions have access to OAAM APIs and objects like session, user, device, location, transaction, logger, and so on as listed in Section 19–4, " Objects Available in Python."

19.6.2 Objects Available in Python

Table 19–4 lists the objects (variables) accessible from Python expressions. For information on available methods, see the corresponding Java interface of each variable in *Oracle Fusion Middleware Java API Reference for Oracle Adaptive Access Manager*.

Table 19–4 Objects Available in Python

Python Variable	Java Interface	Description
oaamAuth	VCryptAuth	OAAM Java API
oaamTracker	VCryptTracker	OAAM Java API
oaamRulesEngine	VCryptRulesEngine	OAAM Java API
oaamCommon	VCryptCommon	OAAM Java API
oaamCC	VCryptCC	OAAM Java API
oaamSession	VCryptTrackerSession	Current session
oaamUser	VCryptTrackerUser	Current session user
oaamDevice	VCryptTrackerDevice	Current session device
oaamLocation	VCryptTrackerLocation	Current session location
oaamTrx	OaamTrxHelper	Current transaction
oaamDb	OaamDbHelper	OAAM DB query interface
oaamLogger	Logger	Logger object for debugging

19.6.3 Examples

Table 19–5 lists a few expressions you can use in the Python Expression condition.

Table 19–5 Python Expressions

Description	Python Expression
Is this the first time the user used this device?	oaamResult = oaamDevice.isUserFirstTime(oaamUser.userId);

Did the current location have more than 5 devices in the past 24 hours?	deviceCount = 5;
	durationUnit = 24;
	durationUnitType = 4; # hours
	authStatus = 0; # success
	oaamResult = oaamLocation.checkDeviceCountMorethan(
	deviceCount, authStatus,
	durationUnit, durationUnitType);

Did the current device have more than 5 users in the past 1 hour?	users = oaamDevice.getAllUsersForDevice(3600); oaamResult = (len(users) > 5);
Does the `AccessType` attribute of the current transaction contain the value `Prescription`?	oaamResult = (oaamTrx.AccessType == "Prescription");
Do the patient and provider entities in the current transaction have the same last name?	oaamResult = (oaamTrx.provider.LastName == oaamTrx.patient.LastName);

Table 19–5 (Cont.) Python Expressions

Description	Python Expression
When the Single-sign-on login-type is `COOKIE_LOGIN`, i,e., `authClientType=1`, check if this is the first time the user used this device	oaamResult = ((oaamSession.authClientType == 11) and oaamDevice.isUserFirstTime(oaamUser.userId));

Did the provider in the current transaction have an appointment with the patient in a given duration?	from jarray import array;
	from java.lang import String;
	from java.lang import Object;
	apptCount = oaamDb.executeSqlSelectSingleValue(
	"select count(*)from oaam_ent_appointmentinfo appt where appt.PatientId=:patientId and appt.ProviderId=:providerId and appt.StartTime between (:trxTime - (:withinHours / 24.0)) and (:trxTime)'",
	array(['patientId', 'providerId', 'trxTime', 'withinHours'], String),
	array([oaamTrx.patient.PatientId, oaamTrx.provider.ProviderId, oaamTrx.createTime, 1], Object));
	oaamResult = (apptCount.intValue() > 0);

Execute user-defined SQL function to determine whether the patient and provider are co-workers	from jarray import array;
	from java.lang import String;
	from java.lang import Object;
	isCoworker = oaamDb.executeSqlSelectSingleValue(
	"select IsSameWorkLocation(:patientID, :providerID) from dual",
	array(['patientID', 'providerID'], String),
	array([oaamTrx.patient.PatientID, oaamTrx.provider.ProviderID], Object));
	oaamResult = (isCoworker.intValue() > 0);

20

Integrating Oracle Access Manager 10*g* and Oracle Adaptive Access Manager 11*g*

Integrating Oracle Adaptive Access Manager (OAAM) with Oracle Access Manager (OAM) enables fine-grain control over the authentication process and provides risk analysis.

This chapter describes the process for integrating Oracle Adaptive Access Manager 11*g* with Oracle Access Manager 10*g*.

It contains the following sections:

20.1 Resource Protection Flow

This section describes the process flow when a user tries to access a protected resource in an Oracle Access Manager and OAAM integration.

1. When a user tries to access a resource protected by Access Manager, he is redirected to the Oracle Adaptive Access Manager login page instead of the Oracle Access Manager login.
2. Oracle Adaptive Access Manager delegates user authentication to Oracle Access Manager.
3. Then, Oracle Adaptive Access Manager performs risk analysis of the user.

20.2 Roadmap for OAAM Integration with Access Manager

Table 20–1 lists the high-level tasks for integrating Oracle Adaptive Access Manager with Access Manager.

Except where specified, the following procedures are required to complete the integration of Oracle Access Adaptive Manager 11*g* and Oracle Access Manager 10*g*.

Table 20–1 Integration Flow for Oracle Access Manager and Oracle Adaptive Access Manager

Number	Task	Information
1	Verify that all required components have been installed and configured prior to integration.	For information, see "Prerequisites".
2	Configure the OAM AccessGate for OAAM Web Server.	For information, see "Configuring OAM AccessGate for OAAM Web Server".
3	Configure the OAM Authentication Scheme.	For information, see "Configuring OAM Authentication Scheme".
4	Configure the Oracle Access Manager connection (optional).	For information, see "Configuring Oracle Access Manager Connection (Optional)".
5	Set up the WebGate for the OAAM web server	For information, see "Setting Up WebGate for OAAM Web Server".
6	Configure the OAM Domain to use OAAM authentication	For information, see "Configuring OAM Domain to Use OAAM Authentication".
7	Configure OHS.	For information, see "Configuring Oracle HTTP Server (OHS)".
8	Configure Oracle Adaptive Access Manager properties.	For information, see "Configuring Oracle Adaptive Access Manager Properties for Oracle Access Manager".
9	Turn off IP validation.	For information, see "Turning Off IP Validation".
10	Validate the Access Manager and Oracle Adaptive Access Manager Integration.	For information, see "Testing Oracle Adaptive Access Manager and Oracle Access Manager Integration".

20.3 Prerequisites

Ensure that the following prerequisites are met before performing the integration:

- All necessary components have been properly installed and configured:
  - Oracle Adaptive Access Manager 11*g*
  - Oracle Access Manager 10.1.4.3
  - Application Server

  For installation information for Oracle Adaptive Access Manager 11*g*, see *Oracle Fusion Middleware Installation Guide for Oracle Identity and Access Management*.

  For installation information for Oracle Access Manager 10*g*, see *Oracle Access Manager Installation Guide 10g (10.1.4.3)*.
- The Oracle Access Manager environment has been configured to protect simple HTML resources using two different authentication schemes:
  - The first authentication scheme uses Basic Over LDAP.

This built-in Web server challenge mechanism requires the user to enter their login ID and password. The credentials supplied are compared to the user's profile in the LDAP directory server.

- The second authentication scheme is a higher-security level and integrates OAAM Server by using a custom form-based authentication scheme.

  This method is similar to the basic challenge method, but users enter information in a custom HTML form. You can choose the information users must provide in the form that you create. A challenge parameter is used. For information about challenge parameters, see "About Challenge Parameters" in Chapter 5, "Configuring User Authentication" in *Oracle Access Manager Access Administration Guide*, 10*g* (10.1.4.3).

  For information on authentication schemes, see Chapter 5, "Configuring User Authentication" in *Oracle Access Manager Access Administration Guide*, 10*g* (10.1.4.3).

20.4 Configuring OAM AccessGate for OAAM Web Server

In Oracle Access Manager and Oracle Adaptive Access Manager integration, the Oracle Access Manager AccessGate fronts the Web server (a traditional WebGate) to OAAM Server. For information on AccessGates, see Chapter 3, "Configuring WebGates and Access Servers" in *Oracle Access Manager Access Administration Guide*, 10*g* (10.1.4.3).

To configure the Oracle Access Manager AccessGate that fronts the Web server to OAAM Server, perform the following steps:

1. Navigate to the Access System Console.

   For information on logging in to the Access System, see Chapter 1, "Preparing for Administration" in *Oracle Access Manager Identity and Common Administration Guide*, 10*g* (10.1.4.3).

2. Click the **Access System Console** link, and then log in as a Master Administrator.

3. Click **Access System Configuration**, then select **Add New AccessGate**.

4. Use the settings in the table below to create a new AccessGate and assign it an Access Server.

   For information on assigning the AccessGate to an Access Server, see Section 3.6, "Associating AccessGates and WebGates with Access Servers," in *Oracle Access Manager Access Administration Guide*, 10*g* (10.1.4.3).

Table 20–2 Oracle HTTP Server (OHS) WebGate Configuration

Parameter	Value	Description
AccessGate Name	ohsWebGate	Name of this AccessGate instance.
Description	AccessGate for Web server hosting OAAM Server	Summary that will help you identify this AccessGate later on.
Hostname	`hostname`	Name or IP address of the server hosting this AccessGate.
Port Number	`port_number`	Web server port protected by the AccessGate when deployed as a WebGate.
AccessGate Password	`passwd`	Password for this AccessGate. The AccessGate uses this password to identity itself to an Access Server.

Table 20–2 (Cont.) Oracle HTTP Server (OHS) WebGate Configuration

Parameter	Value	Description
Debug	<Off>	Off so debug messages between the AccessGate and Access Server are not written.
Maximum user session time (seconds)	3600	Maximum amount of time, in seconds, that a user's authentication session is valid, regardless of their activity. At the expiration of this session time, the user is re-challenged for authentication.
Idle Session Time (seconds)	3600	Amount of time in seconds that a user's authentication session remains valid without accessing any AccessGate protected resources.
Maximum Connections	1	Maximum number of connections this AccessGate can establish with associated Access Servers.
Transport Security	<Open>	Method for encrypting messages between this AccessGate and the Access Servers it is configured to talk to.
IP Validation	<Off>	Determine if a client IP address is the same as the IP address stored in the ObSSOCookie generated for single sign-on.
IP Validation Exception	leave blank	IP addresses to exclude from IP address validation.
Maximum Client Session Time (hours)	24	Connection maintained to the Access Server by the AccessGate.
Failover Threshold	1	Number representing the point when this AccessGate opens connections to secondary Access Servers.
Access server timeout threshold	leave blank	Time (in seconds) during which the AccessGate must wait for a response from the Access Server.
Sleep for (seconds)	60	Number (in seconds) that represents how often this AccessGate checks its connections to Access Servers.
Maximum elements in cache	10000	Maximum number of elements that can be maintained in the URL and authentication scheme caches.
Cache timeout (seconds)	1800	Time period during which cached information remains in the AccessGate cache when neither used nor referenced.
Impersonation Username	leave blank	Name of the trusted user that you created to be used for impersonations.
Impersonation Password	leave blank	Password for the impersonation user name.
Access Management Service	<On>	Whether the Access Management Service is On or Off. On if the Access Server is associated and communicating with AccessGates (which communicate using APIs in the SDK).
Primary HTTP Cookie Domain	*domain_name*	Describes the Web server domain on which the AccessGate is deployed.

Table 20–2 (Cont.) Oracle HTTP Server (OHS) WebGate Configuration

Parameter	Value	Description
Preferred HTTP Host	*hostname*:*port_number*	determines how the host name appears in all HTTP requests as they attempt to access the protected Web server.
Deny on not protected	<Off>	True denies all access to resources on the Web server protected by WebGate unless access is allowed by a policy.
CachePragmaHeader	no-cache	By default, CachePragmaHeader and CacheControlHeader are set to no-cache. This prevents WebGate from caching data at the Web server application and the user's browser.
CacheControlHeader	no-cache	By default, CachePragmaHeader and CacheControlHeader are set to no-cache. This prevents WebGate from caching data at the Web server application and the user's browser.
LogOutURLs	leave blank	Enables you to configure one or more specific URLs that log out a user.
User Defined Parameters	leave blank	Configure the WebGate to work with particular browsers, proxies, and so on.
Assign An Access Server (Primary)	*oam_hostname*:*port_number*	Access server.
Number of Connections	1	Number of connections to the Access Server.

5. Click **AccessGate Configuration**.
6. Click **OK** to search for all AccessGates.

   The new AccessGate is now listed

20.5 Configuring OAM Authentication Scheme

To leverage OAAM Server as an authentication mechanism, Oracle Access Manager must have a defined Authentication Scheme to understand how to direct authentications to OAAM Server. For information on authentication schemes, see Chapter 5, "Configuring User Authentication" in Oracle Access Manager Access Administration Guide, 10*g* (10.1.4.3)

To define the authentication scheme for Oracle Adaptive Access Manager, follow the steps below:

1. From the Access System Console, click the **Access System Configuration** tab.
2. Click **Authentication Management** in the left navigation pane.
3. Click **New**.
4. Using the settings in the table below, begin creating the new OAAM Server authentication scheme:

Table 20–3 OAAM Server Authentication Scheme Configuration

Parameter	Value	Description
Name	Adaptive Strong Authentication	Unique name for the scheme.
Description	Oracle Adaptive Access Manager-OAAM Server virtual authentication pad authentication scheme	Brief description of what the scheme does.
Level	3	Security level of the authentication scheme. The security level of the scheme reflects the challenge method and degree of security used to protect transport of credentials from the user.
Challenge Method	Form	Specifies how authentication is to be performed and the information required to authenticate the user.
Challenge Parameter(s)	form:/oaam_server/oamLoginPage.jsp	Provides WebGate with additional information to perform an authentication form - Indicates where the HTML form is located relative to the host's document directory.
	creds:userid password	Provides WebGate with additional information to perform an authentication creds- Lists all fields used for login in the HTML form.
	action:/oaam_server/	Provides WebGate with additional information to perform an authentication action- URL that the HTML form is posting to.
SSL Required	<No>	Whether users must be authenticated using a server enabled for Secure Sockets Layer (SSL).
Challenge Redirect	`Redirect Url`	URL of another server to which you want to redirect this request if authentication does not take place on the resource Web server.
Enabled	<Disabled/Greyed Out>	Enable or disable the authentication scheme.

5. Click **Save**. The Details for Authentication Scheme display page appears. This page displays the information you entered for the new authentication scheme.
6. Click **Ok** to confirm the saved operation.
7. Select the **Plugins** tab to display the plug-ins for this authentication scheme.
8. Click **Modify**. The Plugins for Authentication Scheme page changes to include the **Add** and **Delete** buttons as well as the **Update Cache** checkbox.
9. Click **Add**. The page changes to include a list of options and a text box for selecting and defining the plug-in to be added.
10. Create the plugin configurations using the information presented in the table below.

Table 20–4 OAAM Server Authentication Scheme Configuration Plugins

Plugin Name	Plugin Parameters
credential_mapping	obMappingBase="dc=<domain>,dc=com",obMappingFilter="(uid=%userid%)"
validate_password	obCredentialPassword="password"

The credential_mapping plug-in maps the user ID to a valid distinguished name (DN) in the directory.

The validate_password plug-in is used to validate the user's password against the LDAP data source.

11. Click **Save**.
12. Click **General**.
13. Click **Modify**.
14. Set **Enabled** to **Yes**.
15. Click **Save**.

20.6 Configuring Oracle Access Manager Connection (Optional)

The AccessGates used by OAAM Server must have host identifier entries. Use the Host Identifiers feature to enter the official name for the host, and every other name by which the host can be addressed by users.

A request sent to any address on the list is mapped to the official hostname, and applicable rules and policies are implemented. This is primarily used in virtual site hosting environments.

For information on configuring host identifiers, see Section 3.7.2, "Configuring Host Identifiers" in Chapter 3, "Configuring WebGates and Access Servers" of *Oracle Access Manager Access Administration Guide*, 10*g* (10.1.4.3).

20.7 Setting Up WebGate for OAAM Web Server

To correctly handle the cookies for authentication and the required HTTP headers for the OAAM Server, OAAM Server must be protected with a standard WebGate and Web server.

To set up the WebGate for use with OAAM Server:

1. Stop the application server (and Web server).
2. Run the WebGate installation program.

For the WebGate configuration, use the following settings:

Table 20–5 Setting Up the WebGate for Use with OAAM Server

Attribute	Value	Description
WebGate ID	ohsWebGate	Unique ID specified in the Access System Console.
WebGate Password	*password*	Password you defined in the Access System Console.

Table 20–5 (Cont.) Setting Up the WebGate for Use with OAAM Server

Attribute	Value	Description
Access Server ID	`Access ServerId`	Access Server ID associated with this WebGate.
DNS Hostname	`OAAM_hostname`	For the Access Server associated with this WebGate.
Port Number	`Access_Server_port_number`	On which the Access Server listens for this WebGate.

For detailed information, refer to Section 9.5.3, "Specifying WebGate Configuration Details" in *Oracle Access Manager Installation Guide 10g (10.1.4.3)* and Chapter 2, "Integrating Oracle HTTP Server" in *Oracle Access Manager Integration Guide 10g (10.1.4.3)*.

20.8 Configuring OAM Domain to Use OAAM Authentication

The OAAM Server authentication should now be operable for Oracle Access Manager policy domains.

To modify the Oracle Access Manager policy domain to use the OAAM authentication scheme (Strong Authentication), follow these steps:

1. In the Access System Console, click the link for the **Policy Manager** at the top of the page.
2. Click **My Policy Domains** in the left navigation pane. A list of policy domains appears.
3. Click the link for the policy domain that you want to view. The General page for the selected policy domain appears.
4. Click **Default Rules**. The General page for the Authentication Rule tab appears. It shows the current configuration for the rule.
5. Click **Modify**. The General page, whose fields you can modify, appears.
6. From the Authentication Scheme drop-down selector, select **Adaptive Strong Authentication**.
7. Click **OK** to confirm the change in authentication schemes.
8. Ensure that **Update Cache** is checked.
9. Click **Save** to save your changes.
10. Close Internet Explorer.

For information on modifying an Authentication Rule for a Policy Domain, see Section 5.9.2, "Modifying an Authentication Rule for a Policy Domain" in Chapter 5, "Configuring User Authentication" of *Oracle Access Manager Access Administration Guide*, 10*g* (10.1.4.3).

20.9 Configuring Oracle HTTP Server (OHS)

mod_wl_ohs is the plug-in for proxying requests from Oracle HTTP Server to Oracle WebLogic server. The mod_wl_ohs module is included in the Oracle HTTP Server installation. You need not download and install it separately. Configure OHS such that it proxies OAAM server. In 11g OHS, that is done by modifying the `mod_wl_ohs.conf` file.

To set up the proxy:

1. Locate the `mod_wl_ohs.conf` file.

   The `mod_wl_ohs.conf` file is located in the following directory:

   ORACLE_INSTANCE/config/OHS/*component_name*

2. Open the `mod_wl_ohs.conf` file and add an entry similar to the following example:

```
<Location /oaam_server>
SetHandler weblogic-handler
WebLogicHost name.mycompany.com
WebLogicPort 24300
</Location>
```

20.10 Configuring Oracle Adaptive Access Manager Properties for Oracle Access Manager

Setting Oracle Adaptive Access Manager properties for Oracle Access Manager and Oracle Access Manager credentials in the Credential Store Framework (CSF) is required for this integration to work.

20.10.1 Setting Oracle Adaptive Access Manager Properties for Oracle Access Manager

Note: Before doing this procedure, you must take into account whether the OAAM Admin Console is being protected.

- If protecting the console, you must take care of user and group creation in the external LDAP store. For details, see Creating Oracle Adaptive Access Manager Administrative Groups and User in LDAP in the *Oracle Fusion Middleware Enterprise Deployment Guide for Oracle Identity Management*.

  OR

- If not protecting the OAAM Admin Console, then the user must be created in the WebLogic Administration Console.

  (Note: You can disable OAAM Admin Console protection by setting the environment variable or Java property WLSAGENT_DISABLED=true.)

To set Oracle Adaptive Access Manager properties for Oracle Access Manager:

1. Start the managed server hosting the Oracle Adaptive Access Manager server.
2. Navigate to the Oracle Adaptive Access Manager Admin Console at http://*oaam_managed_server_host*:*oaam_admin_server_port*/oaam_admin.
3. Log in as a user with access to the property editor.
4. Open the Oracle Adaptive Access Manager property editor to set the Oracle Access Manager properties.

   If a property does not exist, you must add it.

   For the following properties, set the values according to your deployment:

Table 20–6 Configuring Oracle Access Manager Property Values

Property Name	Property Values
bharosa.uio.default.password.auth.provider.classname	com.bharosa.vcrypt.services.OAMOAAMAuthProvider
bharosa.uio.default.is_oam_integrated	true
oracle.oaam.httputil.usecookieapi	true
oaam.uio.oam.host	Access Server host machine name For example, host.example.com
oaam.uio.oam.port	Access Server Port; for example, 3004
oaam.uio.oam.obsso_cookie_domain	Cookie domain defined in Access Server WebGate Agent
oaam.uio.oam.java_agent.enabled	false
oaam.uio.oam.webgate_id	Webgate ID configured in Section 20.4, "Configuring OAM AccessGate for OAAM Web Server."
oaam.uio.login.page	/oamLoginPage.jsp
oaam.uio.oam.authenticate.withoutsession	false
oaam.uio.oam.secondary.host	Name of the secondary Access Server host machine. The property must be added, as it is not set by default. This property is used for high availability. You can specify the fail-over hostname using this property.
oaam.uio.oam.secondary.host.port	Port number of the secondary Access Server The property must be added as it is not set by default. This property is used for high availability. You can specify the fail-over port using this property.
oaam.oam.csf.credentials.enabled	true This property enables configuring credentials in the Credential Store Framework instead of maintaining them using the properties editor. This step is performed so that credentials can be securely stored in CSF.

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*.

20.10.2 Setting Oracle Access Manager Credentials in Credential Store Framework

So that Oracle Access Manager WebGate credentials can be securely stored in the Credential Store Framework, follow these steps to add a password credential to the Oracle Adaptive Access Manager domain:

1. Navigate to the Oracle Fusion Middleware Enterprise Manager Console at http://*weblogic_server_host*:*admin_port*/em.
2. Log in as a WebLogic Administrator.
3. Expand **Base_Domain** in the navigation tree in the left pane.
4. Select your domain name, right-click, select the menu option **Security**, and then select the option **Credentials** in the sub-menu.
5. Click **Create Map**.
6. Click **oaam** to select the map, then click **Create Key**.

7. In the pop-up window make sure Select Map is **oaam**.
8. Provide the following properties and click **OK**.

Table 20–7 Adding Password Credentials to OAAM Domain

Name	Value
Map Name	oaam
Key Name	oam.credentials
Key Type	Password
UserName	Oracle Access Manager user with Administrator rights
Password	Password of Oracle Access Manager WebGate Agent

20.11 Turning Off IP Validation

In order for Oracle Adaptive Access Manager to direct the user to the protected URL after authentication, you must turn off IP validation. For information on configuring IP validation, see Section 3.5.3, "Configuring IP Address Validation for WebGates" in Chapter 3, "Configuring WebGates and Access Servers" in *Oracle Access Manager Access Administration Guide*, 10*g* (10.1.4.3).

To turn off IP validation, follow the steps below:

1. On the Access System main page, click the **Access System Console** link, and then log in as an administrator.
2. On the Access System Console main page, click **Access System Configuration**, and then click the **Access Gate Configuration** link on the left pane to display the AccessGates Search page.
3. Enter the proper search criteria and click **Go** to display a list of AccessGates.
4. Select the AccessGate.

   For example, **ohsWebGate**.
5. Click **Modify** at the bottom of the page.
6. Set **IP Validation** to **off**.
7. Click **Save** at the bottom of the page.

20.12 Testing Oracle Adaptive Access Manager and Oracle Access Manager Integration

To test the configuration, try accessing your application. The Oracle Access Manager will intercept your un-authenticated request and redirect you to OAAM Server to challenge for credentials.

Part V

Custom Development

Part V contains the following chapters:

21

Using and Developing a Virtual Authentication Devices

OAAM includes a suite of highly secure virtual authentication devices as samples to deploy if you choose to. Alteration of these samples is considered custom development. Source art and information in this chapter are provided as a reference to allow you to develop your own custom virtual authentication devices.

> **Note:** These samples are provided in English only.

This chapter contains the following sections:

21.1 About Virtual Authentication Devices

Virtual authentication devices are authenticator interfaces used to protect end users during the process of entering and transmitting authentication credentials and provide them with verification they are authenticating on the valid application. There are many security technologies employed in the authenticator user interfaces. Each virtual authentication device has its own unique set of security features that makes it much more than just an image on a web page.

21.1.1 Virtual Authentication Device Terminology

This section defines terms used in this chapter.

Table 21–1 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.
Sub-jitter	After jitter is calculated each individual key is moved.
Offset	The act of moving a whole key set on screen.
Key Randomization	The act of randomizing the key order. (Scramble)
Timestamp	A string generated from the current system time or client side time.
Masking	Replacing characters in an HTML input field.

21.1.2 Virtual Authentication Device Types

Virtual authentication devices protect users from phishing attacks, data theft, and bots. Each user has an image and a phrase that are used as a shared secret between the business and the end user. The shared secret authenticates the website to the end user, which helps to protect end users from Phishing operations trying to fool them with social engineering.

Figure 21–1 Personalization

Each time PinPad or KeyPad is used the data sent over the wire is random. The actual credential is not entered and sent by the end user. Instead, what is sent are screen coordinates. Basic jitter, sub-jitter and scramble are available. The following subsections introduces you to the virtual authentication devices.

21.1.2.1 TextPad

TextPad is a personalized device that consists of a single form field for entering a password or PIN using a regular keyboard. This method of data entry helps to defend

against phishing primarily. The field can act as a password HTML control that masks data entry. TextPad is often deployed as the default for all users in a large deployment. Then, each user individually can upgrade to another device if desired. The personalized image and phrase a user registers and sees every time the user 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 21–2.

Figure 21–2 TextPad

21.1.2.2 PinPad and KeyPad

PinPad and KeyPad are indirect authentication credential entry virtual devices. They can be invoked at the time of login or in-session if required. A user navigates using their mouse to click the visual "keys." On the wire, the data entered is a string of random numbers that only the OAAM server can decode into the valid password/PIN/data. A configurable number of randomization mechanisms control the balance of usability with the level of required strength. The PinPad and KeyPad are generally given as an optional upgrade users can choose to use or not. This flow ensures only users who want the extra protection utilize it since there is a slight learning curve related to navigation.

PinPad is a lightweight authentication device for entering a numeric PIN. Data input is limited to numerals. It supports key jitter, randomization, and offset.

An example PinPad is shown in Figure 21–3.

Figure 21–3 PinPad

KeyPad is a personalized graphics keyboard. The user uses KeyPad to enter alphanumeric and special character 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 21–4.

Figure 21–4 KeyPad

21.1.2.3 QuestionPad

QuestionPad is a personalized device that renders text in the form of a prompt or question. The user can provide information or an answer for the question 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 21–5.

Figure 21–5 QuestionPad

21.1.3 Virtual Authentication Device Configuration Files and Properties

Text based property files on the server side control how the virtual authentication devices are rendered and how they behave. These files are in the business application for Native deployments or in an application for UIO deployments. Details on the virtual authentication device properties are provided in this chapter for your reference.

21.1.3.1 Files Used in Virtual Authentication Device Configuration

Virtual authentication devices uses the following files:

- **oaam_custom.properties** is the file where custom properties are added for virtual authentication devices, KeySet definitions used in the KeyPad and PinPad devices, and configuration properties that are not localized (translated).
- **oaam_custom_*locale*.properties** are files the administrator customizing the application creates 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 `oaam_custom_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`).

21.1.3.2 Virtual Authentication Device Property Construction

Properties are constructed in the following manner.

`bharosa.authentipad.`*`padtype.property.subproperty=value`*

For example:

```
bharosa.authentipad.textpad.datafield.x=100
```

The pad type values are:

- `textpad`
- `keypad`
- `pinpad`
- `questionpad`

Any defined property can be overwritten or updated by redefining the property in the `oaam_custom.properties` file. This allows only the relevant properties to be changed without having to rewrite all properties in a new set.

21.1.3.3 Randomization and Jitter Properties

Each time PinPad, KeyPad, and QuestionPad are used the data sent over the wire is random. The actual credential is not entered and sent by the end user. Instead, what is sent are screen coordinates or offset values. In addition to basic jitter, sub-jitter and scramble are available through properties. The subsequent figure illustrates how jittering is achieved.

Figure 21–6 Randomization and Jitter

21.1.3.3.1 TextPad Randomization and Jitter Properties

```
bharosa.authentipad.textpad.encrypt.checksum = true
#This contains the values for the checksum encryption process
bharosa.authentipad.textpad.checksum.min = 100
```

```
bharosa.authentipad.textpad.checksum.max = 200
bharosa.authentipad.textpad.checksum.total = 300
```

21.1.3.3.2 KeyPad Randomization and Jitter Properties

```
bharosa.authentipad.keypad.encrypt.jitter = true
bharosa.authentipad.keypad.randomizeKeys=false
bharosa.authentipad.keypad.keyWidthJitter=50
bharosa.authentipad.keypad.keyHeightJitter=15
bharosa.authentipad.keypad.encrypt.checksum = true
bharosa.authentipad.keypad.checksum.min = 100
bharosa.authentipad.keypad.checksum.max = 200
bharosa.authentipad.keypad.checksum.total = 300
```

21.1.3.3.3 PinPad Randomization and Jitter Properties

```
bharosa.authentipad.pinpad.encrypt.jitter = true
bharosa.authentipad.pinpad.randomizeKeys=false
bharosa.authentipad.pinpad.keyWidthJitter=50
bharosa.authentipad.pinpad.keyHeightJitter=15
bharosa.authentipad.pinpad.encrypt.checksum = true
#This contains the values for the checksum encryption process
bharosa.authentipad.pinpad.checksum.min = 100
bharosa.authentipad.pinpad.checksum.max = 200
bharosa.authentipad.pinpad.checksum.total = 300
```

21.1.3.3.4 QuestionPad Randomization and Jitter Properties

```
bharosa.authentipad.questionpad.encrypt.checksum = true
#This contains the values for the checksum encryption process
bharosa.authentipad.questionpad.checksum.min = 100
bharosa.authentipad.questionpad.checksum.max = 200
bharosa.authentipad.questionpad.checksum.total = 300
```

21.2 What Elements of the Authenticator Can Be Customized?

Specific elements of the Authenticator interfaces may be customized. The following are details on these configurations.

21.2.1 Virtual Authentication Device Composition

A virtual authentication device is composed of many elements. Table 21–2 shows the elements which are combined at run time to produce the virtual authentication device for display on the client side.

Table 21–2 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.

21.2.2 Personalized Image

There are 8,423 personalization images for each virtual authentication device. 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=absolute_folder_path_where_oaam_images_are_available
```

To develop custom background images for the virtual authentication devices, perform the following steps:

1. Process the images to the correct resolution for each virtual authentication device being used.

   You can configure a graphic editor to transform the images in batches.

2. Add the images to the correct directories for each virtual authentication device. For example, the TextPad images is placed into the `textpad` folder.

3. For each virtual authentication device, set the following property:

   ```
   bharosa.image.dirlist=absolute_folder_path_where_oaam_images_are_available
   ```

   For example:

   ```
   bharosa.image.dirlist=/scratch/user/Oracle/Middleware/Oracle_IDM1/oaam/
   oaam_images/virtual_authentication_device
   ```

   where *virtual_authentication_device* is one of the subdirectories: `keypad`, `questionpad`, or `textpad`.

21.2.3 Frames

Each of the authenticator interfaces, such as TextPad, KeyPad, PinPad, and so on, has a frame. The frame marks the outer boundary of the authenticator user interface and delineates the virtual authentication device from the rest of the page.

The frame must always be apparent regardless of the graphical treatment to preserve the appearance of a device. The frame may not blend into the surrounding elements of an HTML page to the point were it disappears visually.

The overall size and aspect of each pad is fixed and may not be altered. All elements of the interface must be contained within the frame.

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, but any and all alterations to these images or the properties that correspond to them are considered custom development.

A set of sample background 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. These elements include buttons, fields, personalized phrase and personalized image.

A single image file contains the branding, frame and button images. Some issues to be careful of are text, hot spot, and key sizes. It is not recommended that these be made smaller than the provided samples. Also, there must be an identically named version of each image for each virtual authentication device used in your deployment.

The frame may be altered only in the following ways:

- Colors may be altered for the outline and fill of the frame
- Colors of the buttons on the frame may be altered
- Branding may be altered

> **Note:** If the default value is in a "resource" file, you must specify the override value in `client_resource.properties`.

21.2.3.1 TextPad Authenticator Image and Frame Properties

Table 21–2 lists the TextPad Authenticator Properties

Table 21–3 TextPad Authenticator Properties

Feature	Property
Default background graphic (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) This property applies to the registration page.	`bharosa.uio.`*`appId`*`.register.DeviceTextPad.frame = textpad_bg/TP_O_preview.png`
User Preferences Frame File (Can be application specific) This property applies to the self-service user preferences page.	`bharosa.uio.`*`appId`*`.userpreferences.DeviceTextPad.frame = textpad_bg/TP_O_preview.png`

21.2.3.2 PinPad Authenticator Image and Frame Properties

Table 21–3 lists the PinPad Authenticator Properties

Table 21–4 PinPad Authenticator Properties

Feature	Property
Default background graphic (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

21.2.3.3 QuestionPad Authenticator Image and Frame Properties

Table 21–5 lists the QuestionPad Authenticator Properties

Table 21–5 QuestionPad Authenticator Properties

Feature	Property
Default background graphic (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)

21.2.3.4 KeyPad Authenticator Image and Frame Properties

Table 21–4 lists the KeyPad Authenticator Properties

Table 21–6 KeyPad Authenticator Properties

Feature	Property
Default background graphic (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: *challengeType* 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

21.2.4 Enter Key, Personalized Phrase, and Time Stamp Positioning

Each virtual authentication device has its own unique security features. The default positioning of these elements are provided for your reference.

Table 21–7 Unique Security Features

Visual Element	Description
Enter Key Hotspot	The link area which allows user to submit data entered in the authentication device
Phrase	The personalized phrase assigned to the user at the time of registration. The phrase allows the user to ensure they are on their intended website
Timestamp	The timestamp of when the image was generated, allowing the user to ensure the authentication device is current

21.2.4.1 TextPad Visual Elements

This section provides information on the visual elements of TextPad.

Phrase (Caption)

```
bharosa.authentipad.textpad.caption.personalize = true
bharosa.authentipad.textpad.caption.x = 14
bharosa.authentipad.textpad.caption.y = 203
bharosa.authentipad.textpad.caption.frame = false
bharosa.authentipad.textpad.caption.wrap = false
bharosa.authentipad.textpad.caption.width = 130
bharosa.authentipad.textpad.caption.height = 16
bharosa.authentipad.textpad.caption.font.name = Arial
bharosa.authentipad.textpad.caption.font.color = 000000
bharosa.authentipad.textpad.caption.font.type= 0
bharosa.authentipad.textpad.caption.font.size = 9
```

Timestamp

```
bharosa.authentipad.textpad.timestamp.x = 25
bharosa.authentipad.textpad.timestamp.y = 165
bharosa.authentipad.textpad.timestamp.width = 132
bharosa.authentipad.textpad.timestamp.height = 16
bharosa.authentipad.textpad.timestamp.frame = false
bharosa.authentipad.textpad.timestamp.wrap = false
bharosa.authentipad.textpad.timestamp.font.name = Arial
bharosa.authentipad.textpad.timestamp.font.color = ffffff
bharosa.authentipad.textpad.timestamp.font.type= 0
bharosa.authentipad.textpad.timestamp.font.size = 9
```

Enter Key Hotspot

```
bharosa.authentipad.textpad.enterkey.x=98
bharosa.authentipad.textpad.enterkey.y=181
bharosa.authentipad.textpad.enterkey.width=45
bharosa.authentipad.textpad.enterkey.height=19
bharosa.authentipad.textpad.enterkey.label=enter
bharosa.authentipad.textpad.enterkey.enable=true
```

21.2.4.2 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
```

21.2.4.3 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 `oaam_custom.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.

21.2.4.4 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
```

21.2.4.5 Configuring Text Size for Apple iPhone

To change the TextPad password and QuestionPad answer font size so that it is optimal for the Apple iPhone, add the properties to the `client_resource.properties` file:

```
bharosa.authentipad.textpad.datafield.font.size=12
bharosa.authentipad.questionpad.datafield.font.size=12
```

21.2.5 KeysSets

A KeySet is the configuration that defines what character keys are present on the virtual authentication device. KeySets are used by the KeyPad and PinPad virtual authentication devices.

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.pinpad.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.pinpad.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.pinpad.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.pinpad.default.keyset.row4.enum
bharosa.authentipad.pinpad.default.keyset.enum.row4.order=4
```

Each row is made of the following properties:

Table 21–8 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 21–9 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.

21.3 Customization Steps

The process is as follows:

1. Create a work folder called `oaam_extensions`.

   The folder can be created anywhere if it is outside the installation folder.

2. Extract the `oracle.oaam.extensions.war` file into the work folder.

   In the `oaam_extensions` folder, you should see the following subfolders:

   - `META-INF`
   - `WEB-INF`
   - `WEB-INF\lib`
   - `WEB-INF\classes`

3. Add custom properties to a file named `oaam_custom.properties` and save it to the `oaam_extensions\WEB-INF\classes` directory.

   If your `oaam_custom.properties` is saved in the `oaam_extensions\WEB-INF\classes\bharosa_properties`, you can leave it in that location. You can use either location.

4. Add custom resource bundle values to a file named `client_resource_`*`locale`*`.properties` (where *`locale`* is replaced with relevant locale, such as "en") and save it in the `oaam_extensions\WEB-INF\classes` folder.

User facing text is also considered resource bundle values and you should add these to the file. There are some additional items in OAAM Server such as image paths and regex properties that can be customized based on locale as well.

For example, `bharosa.uio.default.register.DeviceTextPad.frame=textpad_bg/TP_nologo_frame_01.png` and `bharosa.uio.default.DeviceTextPad.default.image = textpad_bg/BG_003.jpg` can be added to this file.

5. Add custom pad related images to `oaam_extensions\WEB-INF\classes\bharosa_properties`.

   If the image exists in the OAAM installation, such as the no logo frame, you do not have to move it to this folder. Only if you are adding a custom file would you need to add it to this folder.

6. Repackage the OAAM Extensions Shared Library, `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/ .
   ```

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

7. Stop all managed servers if they are running.
8. Start the WebLogic Administration Server.
9. Start the Oracle WebLogic Server where Oracle Adaptive Access Manager is deployed and log in to the Oracle WebLogic Administration Console.
10. Remove the `oracle.oaam.extensions.war` currently deployed.
11. Deploy the new `oracle.oaam.extensions.war` file as a shared library with oaam_server and oaam_admin as target applications.

    You may need to target oaam_ offline as well if it is deployed in the same domain.

12. Start all managed servers that are to be used.
13. Test the custom functionality and make sure files added to `oracle.oaam.extensions.war` are used by Oracle Adaptive Access Manager applications.

21.4 Simple Configuration Example

An example is provided for developing an authenticator.

21.4.1 Designing the Frame

The default textpad frame is 148px wide by 223px high as denoted by the properties:

```
bharosa.authentipad.textpad.width=148
bharosa.authentipad.textpad.height=223
```

If you wanted to change these properties to make the new authentipad bigger, you would add these properties with values to the `oaam_custom.properties` file.

The frame itself must have some elements.

- Password entry.
- Enter or Login button.
- A space for the personalized phrase.
- A space for the timestamp.
- A transparent section for the personalized image to show through.

Create a new image that contains all these elements and conforms to width and height pixels.

21.4.2 Positioning the Elements

The elements are set using the properties in Section 21.2.4, "Enter Key, Personalized Phrase, and Time Stamp Positioning". All properties use standard X,Y coordinates from 0,0 in the top left of the image.

21.5 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 Virtual Authentication Device
- Setting Timestamp and Time Zone
- Displaying Virtual Authentication Devices

21.5.1 Setting Up Before Calling the get<pad_type> Method

To get the `bgFile`, you must obtain it from the user by performing:

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

21.5.2 Getting the Virtual Authentication Device

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

You can use the following methods to get commonly used virtual authentication devices:

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

Each method takes the same set of parameters:

Table 21–10 Virtual Authentication Device: Method Parameters

Parameter	Description
String padName	Identifier of the virtual authentication device, 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 virtual authentication device ■ `VCryptLocalizedString(String, VCryptLocale)` ■ `VCryptLocalizedString(String, Locale)` ■ `VCryptLocalizedString(String)`
boolean isADACompliant	Flag to designate if the virtual authentication device 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.

21.5.3 Setting Timestamp and Time Zone

You must set timestamp, time zone and display properties to the virtual authentication device that was obtained.

Table 21–11 describes fields that may need to be set on the virtual authentication device once it is created.

Table 21–11 Virtual Authentication Devices: Setting Additional Fields

Parameter	Description
authentiPad.setTimeStamp(Date timeStamp)	Sets the timestamp to display on the virtual authentication device.
authentiPad.setTimeZone(TimeZone timeZone)	Sets the time zone to display on the virtual authentication device.
authentiPad.setDisplayOnly(boolean displayOnly)	Flag to designate if the virtual authentication device 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.

21.5.4 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`.

21.6 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 through properties.

The accessible versions of the virtual authentication devices contain tabbing, directions and ALT text necessary for navigation through screen reader and other assistive technologies.

To enable these versions, set the `is ADA compliant` flag to true.

For native integration the property to control the virtual authentication devices 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
```

21.7 Customizing the OAAM Server Pages

Areas of the resource bundles you can use to override the page directions and the virtual authentication device text (if desired) are as follows:

 # Username Page

```
bharosa.uio.default.signon.page.title=Sign In:
bharosa.uio.default.signon.page.message=Enter your user name.
```

 # Password Page

```
bharosa.uio.default.password.page.title=Sign In:
bharosa.uio.default.password.page.message=
   Use this security device to enter your password.
bharosa.uio.default.password.page.DeviceHTMLControl.message=Enter your password.
bharosa.uio.default.password.page.DeviceKeyPadFull.message=
   Please use this secure KeyPad to enter your password.
bharosa.uio.default.password.page.DeviceKeyPadAlpha.message=
   Please use this secure KeyPad to enter your password.
bharosa.uio.default.password.page.DeviceTextPad.message=
   Please use this secure TextPad to enter your password.
bharosa.uio.default.password.page.DevicePinPad.message=
   Please use this secure PinPad to enter your PIN.
```

KeyPad Description and Directions

```
bharosa.authentipad.keypad.accessibility.directions =
KeyPad directions: Use the following links to enter your password.
Your personalized caption text comes first,
followed by control links, which are then followed by the key links.
Once you have entered your password, use shift-tab to return
to the enter link to submit your password.

bharosa.authentipad.keypad.security.image.alt = Security Device Image
```

```
bharosa.authentipad.keypad.datafield.label=Password
bharosa.authentipad.keypad.enterkey.label=enter
```

PinPad Description and Directions

```
bharosa.authentipad.pinpad.accessibility.directions =
PinPad directions: Use the following links to enter your numeric pin.
Your personalized caption text comes first, followed by control links,
which are then followed by the numeric links.
Once you have entered your numeric pin,
use shift-tab to return to the enter link to submit your pin.

bharosa.authentipad.pinpad.security.image.alt = Security Device Image
bharosa.authentipad.pinpad.datafield.label=Pin
bharosa.authentipad.pinpad.enterkey.label=enter
```

TextPad Description and Directions

```
bharosa.authentipad.textpad.accessibility.directions =
TextPad directions: Use the following items to validate your device.
Your personalized caption text comes first,
followed by a timestamp to ensure the device was generated for this session.
Once you have entered your password in the previous password entry field,
use tab to navigate to the enter link to submit your password.

bharosa.authentipad.textpad.security.image.alt = Security Device Image
bharosa.authentipad.textpad.datafield.label=Password
bharosa.authentipad.textpad.enterkey.label=enter
```

TextPadReset Description and Directions

```
bharosa.authentipad.textpadreset.accessibility.directions =
TextPad directions:
Use the following items to enter your new password.
Your personalized caption text comes first,
followed by a password input field and password confirmation field.
Once you have entered your password, use tab to navigate to the enter link
to submit your new password.

bharosa.authentipad.textpadreset.security.image.alt = Security Device Image
bharosa.authentipad.textpadreset.datafield.label=Password
bharosa.authentipad.textpadreset.confirmfield.label=Confirm Password
bharosa.authentipad.textpadreset.enterkey.label=enter
```

CaptionPad Description and Directions

```
bharosa.authentipad.captionpad.accessibility.directions =
CaptionPad directions:
Use the following items to enter your new caption text.
Control links come first, followed by a text input field.
Once you have entered your caption, use tab to navigate to the enter link
to submit your new caption.

bharosa.authentipad.captionpad.security.image.alt = Security Device Image
bharosa.authentipad.captionpad.datafield.label=Security Phrase
bharosa.authentipad.captionpad.enterkey.label=enter
```

CaptionPadConfirm Description and Directions

```
bharosa.authentipad.captionconfirmpad.accessibility.directions =
CaptionConfirmPad directions: Use the following items to review your caption text.
To edit your caption text, tab to the Edit link and press Enter.
bharosa.authentipad.captionconfirmpad.security.image.alt = Security Device Image
```

```
bharosa.authentipad.captionconfirmpad.datafield.label=Security Phrase
bharosa.authentipad.captionconfirmpad.enterkey.label=enter
```

QuestionPad Description and Directions

```
bharosa.authentipad.questionpad.accessibility.directions =
QuestionPad directions: Use the following items to enter your answer.
Your personalized question comes first, followed by your personalized caption,
which is then followed by your answer input field.  Once you have entered
your answer, use tab to navigate to the enter link to submit your answer.
bharosa.authentipad.questionpad.security.image.alt = Security Device Image
bharosa.authentipad.questionpad.datafield.label=Answer
bharosa.authentipad.questionpad.enterkey.label=enter
```

21.8 Localizing Virtual Authentication Device in OAAM 11*g*

This section contains the following topics:

- Overview
- Example Using German Locale

21.8.1 Overview

The process is as follows:

1. Create the `oaam_custom_`*`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 `oaam_custom.properties` and save it in the *`temp-folder`*`/WEB-INF/classes` folder. This chapter also contains 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 `oaam_custom.properties`.
6. Deploy the custom OAAM Extensions Shared Library into both the OAAM Managed Servers (OAAM Admin and OAAM Server).
7. Test the localized virtual authentication devices.

21.8.2 Example Using German Locale

The following example shows how to localize the virtual authentication devices in German.

1. Extract the OAAM Extensions shared library WAR file into a temp folder *`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 folder 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 `oaam_custom.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.keypad.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.keypad.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.keypad.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.keypad.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.keypad.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.keypad.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.keypad.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.keypad.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.keypad.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.keypad.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. Repackage the oracle.oaam.extensions.war file 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. 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.

21.9 Changing the Limit of Characters for Passwords

OAAM accepts a limit of 25 characters for passwords. When users logs in to OAAM server for the first time, and the password they enter is more than 25 bytes, they are returned to the username page with an error that their password is invalid.

To change the character limit for passwords entered in to OAAM server, update the value for the following property:

```
bharosa.authentipad.textpad.datafield.maxLength
```

To update the character limit using the OAAM Administration Console, proceed as follows:

1. Log in to the OAAM Administration Console.
2. In the left pane, click **Environment** and double-click **Properties**. The Properties search page is displayed.

3. Search for property with the name `bharosa.authentipad.textpad.datafield.maxLength` and change its value.
4. In cases where the property does not exist, add a new property with the name `bharosa.authentipad.textpad.datafield.maxLength` and the value.

For information on using the shared library to update properties, use this chapter as a reference.

21.10 KeyPad Scenario

A key-logger is software or hardware aimed at stealing any sensitive credentials you enter on your keyboard. The best way to thwart a key-logger is to use another input method for sensitive information. In addition to removing keyboard input from the equation KeyPad also defends against any attacks aiming to steal the data by using a unique data send every session. XY locations are sent rather than the actual data. Thanks to "jitter" the same data will equal a different set of XY coordinates every time the KeyPad is used.

Social engineering attacks are very hard to combat. Personalization consists of a personalized 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 JPEG image served up to the end user for a single use. There is no intelligence to be compromised on the client side.

You could secure a web application without relying on the dependability of the password. You can use all other data present when a user makes an access request to evaluate how risky the situation is. As well, you can issue additional credentials for use only in high-risk situations.

22

Integrating Task Processors

An OAAM sample application is available which demonstrates the use of Integration Processors and Task Processors. To see an example of Java API integration, see Chapter 2, "Natively Integrating Oracle Adaptive Access Manager."

The following sections provide reference information about the OAAM Custom Processor Framework:

- Introduction
- OAAM Sample Framework as a Reference for Integration
- Session Management
- Task Processors
- Challenge Processors
- Checkpoint Processor
- Rules Results Processor
- Integration Processors
- Provider Registration
- Legacy Rules Result Processors

22.1 Introduction

OAAM makes available user flows and interfaces for your business security requirements out of the box, but if your requirement is outside of what is provided by the standard Web application, you can use the OAAM custom processor framework for custom development.

Figure 22–1, "OAAM Standard Web Application" shows the standard web application.

Figure 22–1 OAAM Standard Web Application

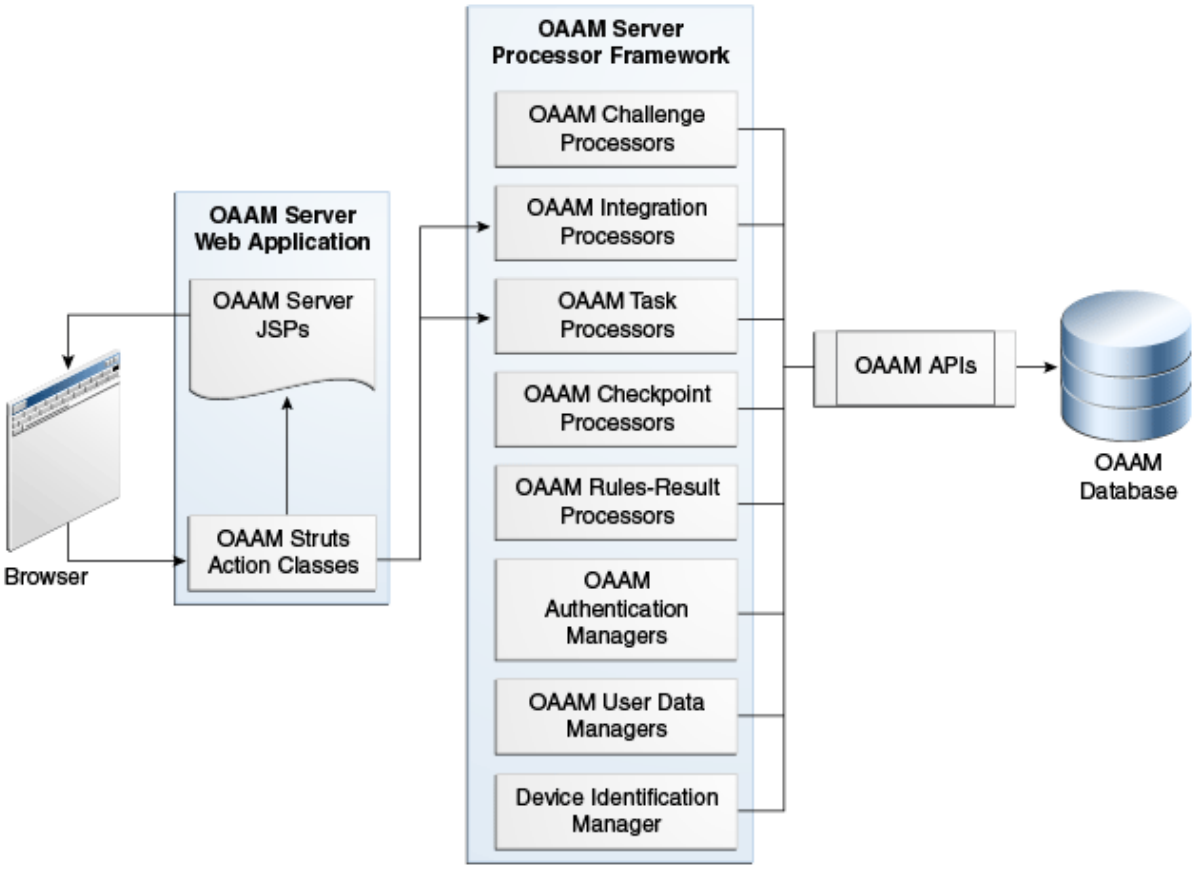
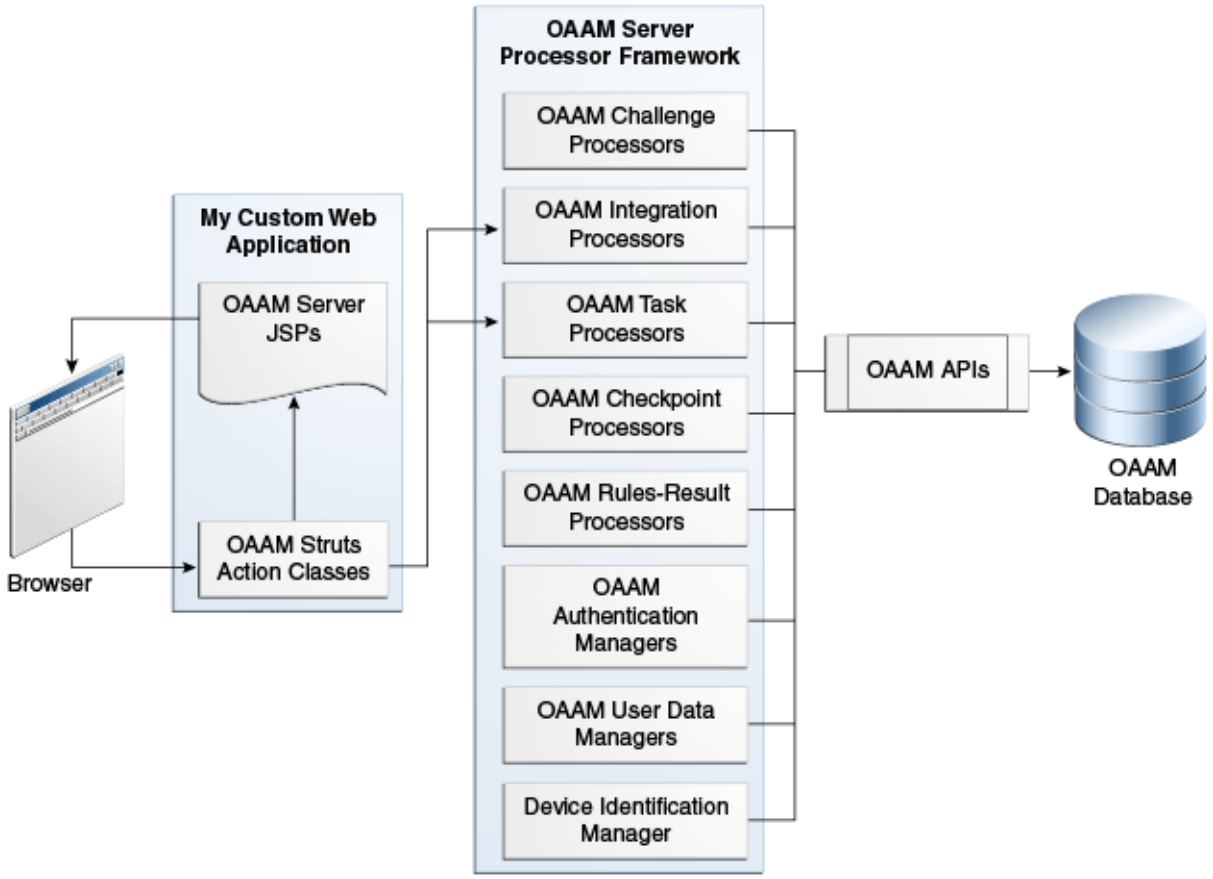


Figure 22–2, "OAAM Processor Framework" shows the different components of the processor framework.

Figure 22–2 OAAM Processor Framework



22.2 OAAM Sample Framework as a Reference for Integration

An OAAM sample framework that illustrates task processor integration is available for your reference. The most recent OAAM sample can be downloaded from My Oracle Support document ID 1501759.1.

The OAAM Sample code is for demonstration purposes to familiarize you with the task processor framework. It is not intended to be used as production code.

> **Note:** This sample does not replace the OAAM sample application for Java API integration available through My Oracle Support document ID 1351899.1. Use either the OAAM sample application that illustrates Java API integration or the one that illustrates task processor integration. The two OAAM sample applications may not be deployed together.

To deploy the sample application, proceed as follows:

1. Create the `oaam_framework_sample` folder.
2. Extract `oaam_framework_sample.war` into `oaam_framework_sample`.
3. Start the WebLogic Server.
4. Navigate to the Oracle WebLogic Administration Console.

http://*oaam_host*:*port*/console

5. Deploy the OAAM Shared Library `$MW_HOME\Oracle_IDM1\oaam\oaam_libs\war\oaam_native_lib.war` as a shared library.
   a. Click **Deployments** under **IAMDomain** (in the left pane).
   b. In the Summary of Deployments page, click the **Install** button.
   c. In the **Path** field, specify `$MW_HOME\Oracle_IDM1\oaam\oaam_libs\war\oaam_native_lib.war`.
   d. Select `oaam_native_lib.war` and click **Next**.
   e. Select the **Install this deployment as a library** option and click **Next**.
   f. In the Select Deployments targets page, select the managed server from the list of servers and click **Next**. Notice the name of the shared library is `oracle.oaam.libs`.

      If the managed server is OAAM Server then there is no need to create a OAAM Data Source. Otherwise create a Data source with JNDI name as `jdbc/OAAM_SERVER_DB_DS` and point it to the OAAM schema.
   g. Click **Finish**.
   h. Click **Activate Changes**.
6. Deploy the OAAM sample application as an application onto the same managed server where the OAAM Shared Library is deployed.
   a. Click **Deployments** under **IAMDomain** (in the left pane).
   b. In the Summary of Deployments page, click the **Install** button.
   c. In the **Path** field, specify the location of the OAAM sample application and click **Next**.
   d. Select the **Install this deployment as an application** option. Click **Next**.
   e. In the Select Deployments targets page, select the managed server from the list of servers and click **Next**.
   f. Click **Finish**.
7. Click **Activate Changes** under the Change Center.
8. In the deployment descriptor file, set the reference to the OAAM shared library `oracle.oaam.libs`.

   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 Oracle WebLogic deployment descriptor file, `weblogic.xml`:

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

   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 Oracle WebLogic deployment descriptor file, `weblogic-application.xml`:

   ```
   <library-ref>
         <library-name>oracle.oaam.libs</library-name>
   </library-ref>
   ```
9. Start the managed server.

10. Navigate to the Oracle WebLogic Administration Console. Click **Lock and Edit** and select the **Deployments** node. On the Summary of Deployments page, find and select the OAAM sample application. Click **Start** > **Servicing all requests**. Click **Yes** to confirm.
11. Log in to OAAM Admin application and import the snapshot.
12. Import the Transaction policies from the `Sample_Txn_Models.zip` file.
13. Navigate to the URL `http://managed_server:port/oaam_framework_sample`. You will see the login page of the OAAM sample application.
14. Enter the user name and then password in the next page. You will go through registration and after that you will see links to Sample Transaction.

Note: The password must be `test` for the initial log in. You must change the password immediately.

22.3 Session Management

`UIOContext` is a context object that maintains thread local variables for the HTTP request and response objects. This allows access to the session (more specifically the `UIOSesisonData` object) always in an application. The context should be populated at the beginning of each request (`HttpServletRequest` and `HttpServletResponse`)

```
UIOContext.setCurrentInstance(new UIOContext(request, response));
```

The UIOSessionData object is maintained in the HTTP session and contains all OAAM session and user data.

22.4 Task Processors

Task processors allow integrators to develop custom java code that perform key OAAM operations.

Task processors:

- Performs a set of OAAM API calls for a specific task
- Updates the OAAM session data
- Separates the user interface from OAAM "work"

Task processors typically have a 1 to 1 relationship with end user pages:

- Login.jsp
  - Presents username field to user
  - Submits username to `loginAction.do`
- LoginAction
  - Accepts data from presentation
  - Maps data to the OAAM task processor
- LoginTaskProcessor
  - Calls the OAAM APIs and updates the UIOSessionData
  - Returns the "target"

22.4.1 Interface and Abstract Class

Each task processor class implements the `TaskProcessorIntf` and extends `AbstractTaskProcessor`.

`TaskProcessorIntf`

- Static variables for task, target, task parameters, and task actions
- Execute method

`AbstractTaskProcessor`

- Breaks the interface execute method into three parts:
  - `preProcess`, `process`, `postProcess`
  - `preProcess` and `postProcess` methods of default classes are empty
  - Allows customizers to extend existing classes to execute code before or after API calls or based on outcomes
- Introduces `validateSession` method to easily validate user session is valid for each task
- Utility methods for determining task action (`show`/ `submit`)

Property Registration

Task processors are registered in the system using properties. The task processor property definition is:

```
bharosa.uio<appId>.task.processor.<taskName> = <java_class>
```

For example:

```
bharosa.uio.default.task.processor.login =
com.bharosa.uio.processor.task.LoginTaskProcessor
```

Accessing a Task Processor

Task processors are accessed by calling the `Utility` method:

```
UIOUtil.getTaskProcessor(UIOSessionData sessionData, String taskId)
```

Creating Parameters

```
TaskParams params = new TaskParams();
params.addParam(LoginTaskProcessor.PARAM_LOGIN_ID, loginId);
params.setActionSubmit();
```

Executing Task

```
target = taskProc.execute(params).getResult();
```

22.4.1.1 TaskProcessorIntf

Each task processor must implement the `TaskProcessorIntf` interface, and it is recommended that it extend an existing task processor or extend the `AbstractTaskprocessor` class. This will allow for customizations to only include

preProcess or postProcess methods if only small additional pieces of java code need to be added for integration purposes.

For example if an external API is required to be called every time LogoutTaskProcessor is called, a CustomLogoutTaskProcessor could be implemented that extends LogoutTaskProcessor and only implements the postProcess method to make the external API call. The property for logout task processor could then be updated to point to the new CustomTaskProcessor class.

```
bharosa.uio.default.task.processor.logout =
com.bharosa.uio.processor.task.CustomLogoutTaskProcessor
```

The TaskProcessorIntf is presented as follows:

```
public interface TaskProcessorIntf {

    public static final String TASK_LOGIN = "login";
    public static final String TASK_JUMP = "jump";
    public static final String TASK_PASSWORD = "password";
    public static final String TASK_CHALLENGE = "challenge";
    public static final String TASK_CHANGE_PASSWORD = "changePassword";
    public static final String TASK_CHANGE_USERNAME = "changeUsername";
    public static final String TASK_FINGERPRINT = "fingerprint";
    public static final String TASK_LOGOUT = "logout";
    public static final String TASK_FORGOT_PASSWORD = "forgotPassword";
    public static final String TASK_CHECKPOINT= "checkpoint";
    public static final String TASK_UPDATE_STATUS = "updateStatus";
    public static final String TASK_REGISTER_SUMMARY = "registerSummary";
    public static final String TASK_REGISTER_IMAGE_PHRASE = "registerImagePhrase";
    public static final String TASK_REGISTER_QUESTIONS = "registerQuestions";
    public static final String TASK_REGISTER_USER_INFO = "registerUserInfo";
    public static final String TASK_REGISTER_COMPLETE = "registerComplete";
    public static final String TASK_REGISTER_SKIP = "registerSkip";
    public static final String TASK_USERPREF_IMAGE_PHRASE =
     "userPrefsImagePhrase";
    public static final String TASK_USERPREF_QUESTIONS = "userPrefsQuestions";
    public static final String TASK_USERPREF_USER_INFO = "userPrefsUserInfo";
    public static final String TASK_USERPREF_COMPLETE = "userPrefsComplete";
    public static final String TASK_TRANSACTION = "transaction";
    public static final String TASK_UPDATE_TRANSACTION_STATUS =
    "updateTransactionStatus";

    public static final String TARGET_ERROR = "error";
    public static final String TARGET_SESSION_EXPIRED = "session_expired";
    public static final String TARGET_FAIL = "fail";
    public static final String TARGET_ACCESS_DENIED = "accessDenied";
    public static final String TARGET_INVALID_USER = "invalid_user";
    public static final String TARGET_SUCCESS = "success";
    public static final String TARGET_UPDATE_STATUS = "updateStatus";
    public static final String TARGET_LOGIN = "login";
    public static final String TARGET_JUMP = "jump";
    public static final String TARGET_RESET_PASSWORD = "resetPassword";
    public static final String TARGET_CHANGE_PASSWORD = "changePassword";
    public static final String TARGET_CHALLENGE= "challenge";
    public static final String TARGET_FORGOT_PASSWORD = "forgotPassword";
    public static final String TARGET_USER_DISABLED = "user_disabled";
    public static final String TARGET_NEXT = "next";


    public static final String TARGET_ALLOW = Preferences.ACTION_ALLOW;
    public static final String TARGET_BLOCK = Preferences.ACTION_BLOCK;
```

```
    public static final String TARGET_REGISTER = Preferences.ACTION_REGISTER;
    public static final String TARGET_REGISTER_SUMMARY = "registerInfo";
    public static final String TARGET_REGISTER_IMAGE_PHRASE =
       "registerAuthenticator";
    public static final String TARGET_REGISTER_QUESTIONS = "registerQuestions";
    public static final String TARGET_REGISTER_QUESTIONS_HTML =
       "registerQuestionsHTML";
    public static final String TARGET_REGISTER_USER_INFO = "registerUserInfo";
    public static final String TARGET_REGISTER_COMPLETE = "registerComplete";
    public static final String TARGET_REGISTER_REQUIRED =
       Preferences.ACTION_REGISTRATION_REQUIRED;
    public static final String PARAM_ACTION = "action";
    public static final String ACTION_SUBMIT = "submitAnswer";
    public static final String ACTION_WAIT = "wait";
    public static final String ACTION_UPGRADE = "upgrade";
    public static final String ACTION_DOWNGRADE = "revert";
    public static final String ACTION_SHOW = "show";

    public String execute(UIOSessionData sessionData, Map<String, String> params);

}
```

22.4.1.2 AbstractTaskProcessor

The `AbstractTaskProcessor` is presented as follows:

```
public abstract class AbstractTaskProcessor implements Serializable,
TaskProcessorIntf {

    public AbstractTaskProcessor() {
    }

    public TaskResult execute(TaskParams params) {
        UIOSessionData sessionData =
UIOContext.getCurrentInstance().getSessionData();

        if (params == null)
            params = new TaskParams();

        String validateSession = validateSession();
        if (validateSession != null) {
            List errorCodes = null;
            if (sessionData != null)
                    errorCodes = sessionData.getErrorCodes();

            return new TaskResult(validateSession, errorCodes);
        }

        String target = preProcess(params);
        if (!StringUtil.equalsIgnoreCase(target, TARGET_ERROR)){
            target = process(params, target);
        }

        target = postProcess(params, target);

        TaskResult result = new TaskResult(target, sessionData.getErrorCodes());
        result.setActionList(sessionData.getActionList());
        result.setChallengeType(UIOUtil.getCurrentChallengeType(sessionData));
        result.setChallengeQuestion(sessionData.getSecretQuestion());
```

```
        result.setRequestId(sessionData.getRequestId());
        result.setLoginId(sessionData.getLoginId());

result.setCurrentAuthDevice(sessionData.getPreferences().getCurrentDevice());

result.setCanSkipRegistration(sessionData.getPreferences().getCanSkipRegistration(
));
        // Update the sessionData object in session for replication.
        UIOSessionData.updateSession();
        return result;
    }

    protected String preProcess(TaskParams params) {
        return null;
    }

    abstract protected String process(TaskParams params, String target);

    protected String postProcess(TaskParams params, String target) {
        return target;
    }

    protected String validateSession(){
        String target = null;
        UIOSessionData sessionData =
 UIOContext.getCurrentInstance().getSessionData();

        if (sessionData == null){
            // Attempt to populate from context.
            sessionData = UIOContext.getCurrentInstance().getSessionData();
            if (sessionData == null){
                logger.info("SessionData is not found in session, so the session
 is expired or UIOContext not pupulated.");
                return TARGET_SESSION_EXPIRED;
            }
        }

        String customerId = sessionData.getCustomerId();
        VCryptAuthUser clientUser = sessionData.getClientAuthUser();

        if (clientUser == null && StringUtil.isEmpty(customerId)) {
            logger.info("Client User is not found in session, so the session is
 expired.");
            target = TARGET_SESSION_EXPIRED;
        } else if (clientUser == null && !StringUtil.isEmpty(customerId)) {
            logger.info("Client User is not found in session.");
            target = TARGET_ACCESS_DENIED;
        }

        return target;

    }

    public boolean isShow(TaskParams params){
        return StringUtil.equalsIgnoreCase(params.getAction(),
 TaskProcessorIntf.ACTION_SHOW);
    }

    public boolean isSubmit(TaskParams params){
        return StringUtil.equalsIgnoreCase(params.getAction(),
```

```
 TaskProcessorIntf.ACTION_SUBMIT);
    }
}
```

22.4.1.3 Default Classes

The following properties define the default task processors:

```
bharosa.uio.default.task.processor.login =
 com.bharosa.uio.processor.task.LoginTaskProcessor
bharosa.uio.default.task.processor.jump =
 com.bharosa.uio.processor.task.JumpTaskProcessor
bharosa.uio.default.task.processor.password =
 com.bharosa.uio.processor.task.PasswordTaskProcessor
bharosa.uio.default.task.processor.challenge =
 com.bharosa.uio.processor.task.ChallengeUserTaskProcessor
bharosa.uio.default.task.processor.changePassword =
 com.bharosa.uio.processor.task.ChangePasswordTaskProcessor
bharosa.uio.default.task.processor.changeUsername =
 com.bharosa.uio.processor.task.ChangeUsernameTaskProcessor
bharosa.uio.default.task.processor.fingerprint =
 com.bharosa.uio.processor.task.FingerprintTaskProcessor
bharosa.uio.default.task.processor.logout =
 com.bharosa.uio.processor.task.LogoutTaskProcessor
bharosa.uio.default.task.processor.forgotPassword =
 com.bharosa.uio.processor.task.ForgotPasswordTaskProcessor
bharosa.uio.default.task.processor.checkpoint =
 com.bharosa.uio.processor.task.CheckpointTaskProcessor
bharosa.uio.default.task.processor.updateStatus =
 com.bharosa.uio.processor.task.UpdateAuthStatusTaskProcessor
bharosa.uio.default.task.processor.registerSummary =
 com.bharosa.uio.processor.task.RegisterSummaryTaskProcessor
bharosa.uio.default.task.processor.registerImagePhrase =
 com.bharosa.uio.processor.task.RegisterImageAndPhraseTaskProcessor
bharosa.uio.default.task.processor.registerQuestions =
 com.bharosa.uio.processor.task.RegisterQuestionsTaskProcessor
bharosa.uio.default.task.processor.registerUserInfo =
 com.bharosa.uio.processor.task.RegisterUserInfoTaskProcessor
bharosa.uio.default.task.processor.registerComplete =
 com.bharosa.uio.processor.task.RegisterCompleteTaskProcessor
bharosa.uio.default.task.processor.registerSkip =
 com.bharosa.uio.processor.task.RegisterSkipTaskProcessor
bharosa.uio.default.task.processor.userPrefsImagePhrase =
 com.bharosa.uio.processor.task.UserPreferencesImageAndPhraseTaskProcessor
bharosa.uio.default.task.processor.userPrefsQuestions =
 com.bharosa.uio.processor.task.UserPreferencesQuestionsTaskProcessor
bharosa.uio.default.task.processor.userPrefsUserInfo =
 com.bharosa.uio.processor.task.UserPreferencesUserInfoTaskProcessor
bharosa.uio.default.task.processor.userPrefsComplete =
 com.bharosa.uio.processor.task.UserPreferencesCompleteTaskProcessor
bharosa.uio.default.task.processor.transaction =
 com.bharosa.uio.processor.task.TransactionTaskProcessor
bharosa.uio.default.task.processor.updateTransactionStatus =
 com.bharosa.uio.processor.task.UpdateTransactionStatusTaskProcessor
```

*denotes optional parameter

Table 22–1 AbstractTaskProcessor

Class name	Params	Results	Notes
LoginTaskProcessor	LoginTaskProcessor.PARAM_LOGIN_ID	TaskProcessorIntf.TARGET_SUCCESS	Load user from login id
	LoginTaskProcessor.PARAM_AUTH_STATUS*	TaskProcessorIntf.TARGET_ERROR	
	Elements defined by user defined enum: bharosa.uio.*appId*.credentials.enum*		
JumpTaskProcessor	JumpTaskProcessor.PARAM_OFFSET	TaskProcessorIntf.TARGET_JUMP	Create OAAM session
	JumpTaskProcessor.PARAM_JUMP	TaskProcessorIntf.TARGET_UPDATE_STATUS	Perform browser fingerprinting
		TaskProcessorIntf.TARGET_SUCCESS	Run pre-authentication rules
		TaskProcessorIntf.TARGET_LOGIN	
		TaskProcessorIntf.TARGET_ERROR	
FingerprintTaskProcessor	(FP data from UIOSesisonData)	TaskProcessorIntf.TARGET_SUCCESS	Process digital fingerprinting (Flash or other external fingerprinting requests). Uses device identification processor.
PasswordTaskProcessor	PasswordTaskProcessor.PARAM_PASSWORD	TaskProcessorIntf.TARGET_SUCCESS	Process password submission to authenticate the user using AuthManager class.
		TaskProcessorIntf.TARGET_RESET_PASSWORD	
		TaskProcessorIntf.TARGET_INVALID_USER	
		TaskProcessorIntf.TARGET_SESSION_EXPIRED	
		TaskProcessorIntf.TARGET_ACCESS_DENIED	
		TaskProcessorIntf.TARGET_FAIL	
		TaskProcessorIntf.TARGET_ERROR	
		TaskProcessorIntf.TARGET_UPDATE_STATUS	

Table 22–1 (Cont.) AbstractTaskProcessor

Class name	Params	Results	Notes
ChallengeUserTaskProcessor	ChallengeUserTaskProcessor.PARAM_ANSWER ChallengeUserTaskProcessor.PARAM_REGISTER_DEVICE	TaskProcessorIntf.TARGET_CHALLENGE TaskProcessorIntf.TARGET_UPDATE_STATUS TaskProcessorIntf.TARGET_SESSION_EXPIRED TaskProcessorIntf.TARGET_ACCESS_DENIED TaskProcessorIntf.TARGET_FAIL TaskProcessorIntf.TARGET_ERROR	Process user challenges using challenge processor classes based on challenge type performed.
UpdateAuthStatusTaskProcessor	UpdateAuthStatusTaskProcessor.PARAM_STATUS UpdateAuthStatusTaskProcessor.PARAM_SECONDARY_GROUP	(Post Authentication Rule Action) TaskProcessorIntf.TARGET_SESSION_EXPIRED TaskProcessorIntf.TARGET_ACCESS_DENIED TaskProcessorIntf.TARGET_FAIL TaskProcessorIntf.TARGET_ERROR	Updates the session status in OAAM
RegisterSummaryTaskProcessor	TaskProcessorIntf.TARGET_SUCCESS TaskProcessorIntf.TARGET_NEXT	Executed when registration summary is requested. (No out of the box functionality)	.
RegisterImageAndPhraseTaskProcessor	TaskProcessorIntf.TARGET_SUCCESS TaskProcessorIntf.TARGET_NEXT TaskProcessorIntf.TARGET_SESSION_EXPIRED TaskProcessorIntf.TARGET_ACCESS_DENIED TaskProcessorIntf.TARGET_ERROR	Processes authenticator registration. Image, Phrase, VAD upgrade/downgrade.	.

Table 22–1 (Cont.) AbstractTaskProcessor

Class name	Params	Results	Notes
RegisterQuestionsTaskProcessor	RegisterQuestionsTaskProcessor.PARAM_QUESTION_COUNT RegisterQuestionsTaskProcessor.PARAM_QUESTION_ID_BASE+# RegisterQuestionsTaskProcessor.PARAM_QUESTION_TEXT_BASE+# RegisterQuestionsTaskProcessor.PARAM_QUESTION_ANSWER_BASE+# RegisterQuestionsTaskProcessor.PARAM_REGISTER_DEVICE*	TaskProcessorIntf.TARGET_SUCCESS TaskProcessorIntf.TARGET_NEXT TaskProcessorIntf.TARGET_SESSION_EXPIRED TaskProcessorIntf.TARGET_ACCESS_DENIED TaskProcessorIntf.TARGET_FAIL TaskProcessorIntf.TARGET_ERROR	Processes and validates challenge question registration.
RegisterUserInfoTaskProcessor	Elements defined by user defined enum: bharosa.uio.*appId*.userinfo.inputs.enum RegisterUserInfoTaskProcessor.PARAM_REGISTER_DEVICE* RegisterUserInfoTaskProcessor.PARAM_OPTOUT*	TaskProcessorIntf.TARGET_SUCCESS TaskProcessorIntf.TARGET_NEXT TaskProcessorIntf.TARGET_SESSION_EXPIRED TaskProcessorIntf.TARGET_ACCESS_DENIED TaskProcessorIntf.TARGET_ERROR	Processes and validates OTP contact information registration.
RegisterCompleteTaskProcessor	TaskProcessorIntf.TARGET_NEXT TaskProcessorIntf.TARGET_SESSION_EXPIRED TaskProcessorIntf.TARGET_ACCESS_DENIED TaskProcessorIntf.TARGET_ERROR	Updates user registration status when registration flow is complete.	.
RegisterSkipTaskProcessor	TaskProcessorIntf.TARGET_SUCCESS TaskProcessorIntf.TARGET_SESSION_EXPIRED TaskProcessorIntf.TARGET_ACCESS_DENIED TaskProcessorIntf.TARGET_ERROR	Skips registration steps if user is allowed to skip registration.	.

Table 22–1 (Cont.) AbstractTaskProcessor

Class name	Params	Results	Notes
UserPreferencesImageAndPhraseTaskProcessor	TaskProcessorIntf.TARGET_SUCCESS TaskProcessorIntf.TARGET_NEXT TaskProcessorIntf.TARGET_SESSION_EXPIRED TaskProcessorIntf.TARGET_ACCESS_DENIED TaskProcessorIntf.TARGET_ERROR	Processes authenticator registration. Image, Phrase, VAD upgrade/downgrade.	.
UserPreferencesQuestionsTaskProcessor	RegisterQuestionsTaskProcessor.PARAM_QUESTION_COUNT RegisterQuestionsTaskProcessor.PARAM_QUESTION_ID_BASE+# RegisterQuestionsTaskProcessor.PARAM_QUESTION_TEXT_BASE+# RegisterQuestionsTaskProcessor.PARAM_QUESTION_ANSWER_BASE+# RegisterQuestionsTaskProcessor.PARAM_REGISTER_DEVICE*	TaskProcessorIntf.TARGET_SUCCESS TaskProcessorIntf.TARGET_NEXT TaskProcessorIntf.TARGET_SESSION_EXPIRED TaskProcessorIntf.TARGET_ACCESS_DENIED TaskProcessorIntf.TARGET_FAIL TaskProcessorIntf.TARGET_ERROR	Processes and validates challenge question registration.
UserPreferencesUserInfoTaskProcessor	Elements defined by user defined enum: bharosa.uio.*appId*.userinfo.inputs.enum RegisterUserInfoTaskProcessor.PARAM_REGISTER_DEVICE* RegisterUserInfoTaskProcessor.PARAM_OPTOUT*	TaskProcessorIntf.TARGET_SUCCESS TaskProcessorIntf.TARGET_NEXT TaskProcessorIntf.TARGET_SESSION_EXPIRED TaskProcessorIntf.TARGET_ACCESS_DENIED TaskProcessorIntf.TARGET_ERROR	Processes and validates OTP contact information registration.
UserPreferencesCompleteTaskProcessor	TaskProcessorIntf.TARGET_NEXT TaskProcessorIntf.TARGET_SESSION_EXPIRED TaskProcessorIntf.TARGET_ACCESS_DENIED TaskProcessorIntf.TARGET_ERROR	Updates user registration status when user preferences flow is complete.	.

Table 22–1 (Cont.) AbstractTaskProcessor

Class name	Params	Results	Notes
ForgotPasswordTaskProcessor	ForgotPasswordTaskProcessor. PARAM_ PAGEID	TaskProcessorIntf.TARGET_FORGOT_PASSWORD TaskProcessorIntf.TARGET_SUCCESS TaskProcessorIntf.TARGET_NO_PROXY TaskProcessorIntf.TARGET_SESSION_EXPIRED TaskProcessorIntf.TARGET_FAIL	.
LogoutTaskProcessor	TaskProcessorIntf.TARGET_SUCCESS	Resets user session.	.
CheckpointTaskProcessor	CheckpointTaskProcessor.PARAM_ CHECKPOINT If not present, will use sessionData.getCheckpoint()	Rule action result TaskProcessorIntf.TARGET_SUCCESS	Runs the checkpoint passed as param or the checkpoint set in sessionData and returns the result.
ChangePasswordTaskProcessor	ChangePasswordTaskProcessor.PARAM_ PAGEID ChangePasswordTaskProcessor.PARAM_ PASSWORD_OLD ChangePasswordTaskProcessor.PARAM_ PASSWORD_NEW ChangePasswordTaskProcessor.PARAM_ PASSWORD_CONFIRM	TaskProcessorIntf.SUCCESS TaskProcessorIntf.TARGET_UPDATE_STATUS TaskProcessorIntf.TARGET_CHANGE_PASSWORD TaskProcessorIntf.TARGET_ACCESS_DENIED TaskProcessorIntf.TARGET_ERROR TaskProcessorIntf.TARGET_ERROR	Uses password manager to update users password.

Table 22–1 (Cont.) AbstractTaskProcessor

Class name	Params	Results	Notes
ChangeUsernameTaskProcessor	ChangeUsernameTaskProcessor. PARAM_ USERNAME_NEW	TaskProcessorIntf.SUCCESS TaskProcessorIntf.TARGET_ACCESS_DENIED TaskProcessorIntf.TARGET_FAIL	Updates a users login id / username.
TransactionTaskProcessor	TransactionTaskProcessor.PARAM_ TRANSACTION_TYPE TransactionTaskProcessor.PARAM_ CHECKPOINT TransactionTaskProcessor.PARAM_EXTERNAL_ TRANSACTION_ID TransactionTaskProcessor.PARAM_AUTO_ UPDATE (optional - prevents status update if set to "false") TransactionTaskProcessor.CHECKPOINT_NONE (optional, prevents checkpoints from running if set to "true")	(Transaction Rule Action) TaskProcessorIntf.TARGET_SESSION_EXPIRED TaskProcessorIntf.TARGET_ACCESS_DENIED TaskProcessorIntf.TARGET_FAIL TaskProcessorIntf.TARGET_ERROR	Runs pre transaction and create transaction checkpoints
UpdateTransactionStatusTaskProcessor	TransactionTaskProcessor.PARAM_ TRANSACTION_TYPE TransactionTaskProcessor.PARAM_ CHECKPOINT TransactionTaskProcessor.PARAM_EXTERNAL_ TRANSACTION_ID TransactionTaskProcessor.PARAM_STATIUS TransactionTaskProcessor.CHECKPOINT_NONE (optional, prevents checkpoints from running if set to "true")	(Transaction Rule Action) TaskProcessorIntf.TARGET_SESSION_EXPIRED TaskProcessorIntf.TARGET_ACCESS_DENIED TaskProcessorIntf.TARGET_FAIL TaskProcessorIntf.TARGET_ERROR	Runs pre-update and update transaction checkpoints.

22.4.2 Task Processor Registration

Task Processors allow for custom java classes to handle OAAM Server user tasks, such as entering username, entering password, or being challenged.

```
bharosa.uio.default.task.processor.login =
 com.bharosa.uio.processor.task.LoginTaskProcessor
bharosa.uio.default.task.processor.jump =
 com.bharosa.uio.processor.task.JumpTaskProcessor
bharosa.uio.default.task.processor.password =
 com.bharosa.uio.processor.task.PasswordTaskProcessor
bharosa.uio.default.task.processor.challenge =
 com.bharosa.uio.processor.task.ChallengeUserTaskProcessor
```

```
bharosa.uio.default.task.processor.changePassword =
 com.bharosa.uio.processor.task.ChangePasswordTaskProcessor
bharosa.uio.default.task.processor.changeUsername =
 com.bharosa.uio.processor.task.ChangeUsernameTaskProcessor
bharosa.uio.default.task.processor.fingerprint =
 com.bharosa.uio.processor.task.FingerprintTaskProcessor
bharosa.uio.default.task.processor.logout =
 com.bharosa.uio.processor.task.LogoutTaskProcessor
bharosa.uio.default.task.processor.forgotPassword =
 com.bharosa.uio.processor.task.ForgotPasswordTaskProcessor
bharosa.uio.default.task.processor.checkpoint =
 com.bharosa.uio.processor.task.CheckpointTaskProcessor
bharosa.uio.default.task.processor.updateStatus =
 com.bharosa.uio.processor.task.UpdateAuthStatusTaskProcessor
bharosa.uio.default.task.processor.registerSummary =
 com.bharosa.uio.processor.task.RegisterSummaryTaskProcessor
bharosa.uio.default.task.processor.registerImagePhrase =
 com.bharosa.uio.processor.task.RegisterImageAndPhraseTaskProcessor
bharosa.uio.default.task.processor.registerQuestions =
 com.bharosa.uio.processor.task.RegisterQuestionsTaskProcessor
bharosa.uio.default.task.processor.registerUserInfo =
 com.bharosa.uio.processor.task.RegisterUserInfoTaskProcessor
bharosa.uio.default.task.processor.registerComplete =
 com.bharosa.uio.processor.task.RegisterCompleteTaskProcessor
bharosa.uio.default.task.processor.registerSkip =
 com.bharosa.uio.processor.task.RegisterSkipTaskProcessor
bharosa.uio.default.task.processor.userPrefsImagePhrase =
 com.bharosa.uio.processor.task.UserPreferencesImageAndPhraseTaskProcessor
bharosa.uio.default.task.processor.userPrefsQuestions =
 com.bharosa.uio.processor.task.UserPreferencesQuestionsTaskProcessor
bharosa.uio.default.task.processor.userPrefsUserInfo =
 com.bharosa.uio.processor.task.UserPreferencesUserInfoTaskProcessor
bharosa.uio.default.task.processor.userPrefsComplete =
 com.bharosa.uio.processor.task.UserPreferencesCompleteTaskProcessor
bharosa.uio.default.task.processor.transaction =
 com.bharosa.uio.processor.task.TransactionTaskProcessor
bharosa.uio.default.task.processor.updateTransactionStatus =
 com.bharosa.uio.processor.task.UpdateTransactionStatusTaskProcessor
```

22.5 Challenge Processors

The OAAM Server provides a challenge processor framework that allows for custom implementations of challenge mechanisms.

22.5.1 What are Challenge Processors

Challenge processors are used for the generation and validation of user challenges. They have the ability to integrate with external challenge services.

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.

A challenge processor is java code that implements the `ChallengeProcessorIntf` interface or extends the `AbstractChallengeProcessor` class.

22.5.2 How to Create Challenge Processors

This section contains information on the challenge processor class and methods to implement. An implementation example is also provided for your reference.

22.5.2.1 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, see Chapter 7, "Using the OAAM Extensions Shared Library to Customize OAAM."

22.5.2.2 Methods

The methods used in a challenge processor are listed in the sections following.

Table 22–2 Challenge Processor Methods

Methods	Description
protected boolean generateSecret(UIOSessionData sessionData, boolean isRetry)	This method used to generate code to send to client
protected boolean validateAnswer(UIOSessionData sessionData, String answer)	This method used to validate the user answer.
public String checkDeliveryStatus(UIOSessionData sessionData, boolean userWaiting, boolean isRetry)	This method used to provide a wait until message is sent.
public boolean isServiceAvailable(UIOSessionData sessionData)	This method used to check if external service is available.

22.5.2.3 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;
  }
}
```

22.5.2.4 Secret (PIN) Implementation

The `AbstractOTPChallengeProcessor` class has a default pin generation method, `generateCode`, that you can override to provide your pin generation logic.

22.5.3 Define the Delivery Channel Types for the Challenge Processors

This section contains instructions on defining a delivery channel type for the challenge processors to use for challenging the user. Examples are provided for your reference.

22.5.3.1 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
```

You use the challenge type enum 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 22.5.2.3, "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" typically refers to the delivery channel used to send to the user.

22.5.3.2 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. An following example shows **ChallengeEmail** as the Challenge Type.

```
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 22–3 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
```

22.5.4 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, see Chapter 7, "Using the OAAM Extensions Shared Library to Customize OAAM."

22.5.4.1 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
```

22.5.4.2 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 22–4 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.manager.user.DefaultContactInfoManager
```

22.5.5 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.

Note: Bypassing the authentipad checkpoint and using properties to determine which virtual device to use will not allow you to use an alternate device for challenges on mobile browsers. Some virtual devices are not ideal for mobile browsing, such as PinPad and KeyPad.

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, 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 22–5 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.

22.6 Checkpoint Processor

Checkpoint processors allow for custom java classes to be executed when a specific checkpoint is run. The properties take the following form:

```
bharosa.uio.appId.checkpoint.processor.checkpoint=
fully_qualified_class_name
```

Checkpoint processors are registered with the application using the following properties:

```
bharosa.uio.default.checkpoint.processor.default =
 com.bharosa.uio.processor.checkpoint.DefaultCheckpointProcessor
bharosa.uio.default.checkpoint.processor.preauth =
 com.bharosa.uio.processor.checkpoint.PreAuthCheckpointProcessor
bharosa.uio.default.checkpoint.processor.postauth =
 com.bharosa.uio.processor.checkpoint.PostAuthCheckpointProcessor
bharosa.uio.default.checkpoint.processor.ChallengeUser =
 com.bharosa.uio.processor.checkpoint.ChallengeCheckpointProcessor
bharosa.uio.default.checkpoint.processor.ForgotPassword =
 com.bharosa.uio.processor.checkpoint.ForgotPasswordCheckpointProcessor
bharosa.uio.default.checkpoint.processor.authentiPad =
 com.bharosa.uio.processor.checkpoint.AuthentiPadCheckpointProcessor
```

If no checkpoint processor is defined, then a default checkpoint processor will be used. The default checkpoint processor is defined by similar property.

```
bharosa.uio.appId.checkpoint.processor.default= fully_qualified_class_name
```

Each checkpoint processor must implement the `CheckpointProcessorIntf` interface.

```
public interface CheckpointProcessorIntf{

  public String processCheckpoint(UIOSessionData sessionData,
VCryptRulesResult rulesResult);

}
```

It is recommended that each custom checkpoint processor extend the `DefaultCheckpointProcessor` class. The default checkpoint processor has some convenience methods to help process the checkpoint.

```
/**
 * Default checkpoint handling
 *
 * Instanciates and executes appropriatlely configured RulesResultProcessors
 */
public class DefaultCheckpointProcessor implements CheckpointProcessorIntf{

  static Logger logger = Logger.getLogger(DefaultCheckpointProcessor.class);

  /**
   * @param sessionData
   * @param rulesResult
   * @return
   */
  protected RulesResultProcessorIntf getResultProcessor(UIOSessionData
 sessionData, VCryptRulesResult rulesResult){
    if (rulesResult == null || rulesResult.getResult() == null){
      logger.info("getResultProcessor :: Rules Result null - Using default rules
 result processor.");
      return getDefaultResultProcessor(sessionData);
```

```
    }

    String resultName = rulesResult.getResult();

    return UIOUtil.getRulesResultProcessor(sessionData, resultName);
  }

  /**
   * @param sessionData
   * @return
   */
  protected RulesResultProcessorIntf getDefaultResultProcessor(UIOSessionData
sessionData){
    return UIOUtil.getRulesResultProcessor(sessionData, "default");
  }

  /**
   * @param checkpointType
   * @return
   */
  protected String getCheckpointId(int checkpointType){
    String checkpointId = null;
    UserDefEnumElement checkpointElem =
UserDefEnum.getElement(BharosaUIOConstants.ENUM_PROFILE_TYPE, checkpointType);
    if (checkpointElem != null)
      checkpointId = checkpointElem.getElementId();

    return checkpointId;
  }

  /**
   * @param sessionData
   * @param rulesResult
   * @return
   */
  public String processCheckpoint(UIOSessionData sessionData, VCryptRulesResult
rulesResult){

    if (logger.isDebugEnabled())
logger.debug("DefaultCheckpointProcessor::processCheckpoint() entered.");

    List actionList = rulesResult.getAllActions();
    sessionData.setActionList(actionList);

    RulesResultProcessorIntf rulesResultProcessor =
getResultProcessor(sessionData, rulesResult);
    if (rulesResultProcessor == null){
      logger.error("processCheckpoint :: rulesResultProcessor is null.");
      return Preferences.ACTION_ERROR;
    }

    return rulesResultProcessor.processRulesResult(sessionData, rulesResult);
  }

}
```

22.7 Rules Results Processor

Rules Result Processors allow for custom java classes to be executed when a specific rule action is returned by the rules engine. The properties take the form:

bharosa.uio.*appId*. rules.result.processor.*rule_action*= *fully_qualified_class_name*

Rule results processors are registered with the application using these properties:

```
bharosa.uio.default.rules.result.processor.default =
 com.bharosa.uio.processor.rules.result.DefaultRulesResultProcessor
bharosa.uio.default.rules.result.processor.Password =
 com.bharosa.uio.processor.rules.result.PasswordRulesResultProcessor
bharosa.uio.default.rules.result.processor.Challenge =
 com.bharosa.uio.processor.rules.result.ChallengeRulesResultProcessor
bharosa.uio.default.rules.result.processor.Register =
 com.bharosa.uio.processor.rules.result.RegisterRulesResultProcessor
bharosa.uio.default.rules.result.processor.RegistrationRequired =
 com.bharosa.uio.processor.rules.result.RegistrationRequiredRulesResultProcessor
```

If no rules result processor is defined, then a default rules result processor will be used. The default rules result processor is defined by similar property.

bharosa.uio.*appId*. rules.result.processor.**default**= *fully_qualified_class_name*

Each rules result processor must implement the `RulesResultProcessorIntf` interface.

```
public interface RulesResultProcessorIntf{

  public String processRulesResult(UIOSessionData sessionData, VCryptRulesResult
 rulesResult);

}
```

It is recommended that each custom checkpoint processor extend the `DefaultRulesResultProcessor` class. The default rules result processor has some convenience methods to help process the rules result.

```
/**
 * Default handling of rule results
 */
public class DefaultRulesResultProcessor implements RulesResultProcessorIntf{

  static Logger logger = Logger.getLogger(DefaultRulesResultProcessor.class);

  /**
   * Process rules results, updating user status as defined by
VCryptRulesResul object.
   *
   * @param sessionData
   * @param rulesResult
   * @return
   */
  public String processRulesResult(UIOSessionData sessionData, VCryptRulesResult
 rulesResult){

    if (logger.isDebugEnabled())
 logger.debug("DefaultRulesResultProcessor::processRulesResult() entered.");
```

```
   if (rulesResult == null)
     return null;

   if (rulesResult.getResult() == null){
     if (logger.isDebugEnabled())
logger.debug("DefaultRulesResultProcessor.processRulesResult(): rules result is
null.");
     return Preferences.ACTION_NONE;
   }

   Preferences prefs = sessionData.getPreferences();
   String ruleAction = rulesResult.getResult();
   int authStatus = sessionData.getAuthResult();
   int newStatus = authStatus;

   UserDefEnumElement actionElement =
UserDefEnum.getElement(IBharosaConstants.ENUM_RULE_ACTION_ID, ruleAction);
   String overRideStatus = actionElement.getProperty("authStatus");

   if (!StringUtil.isEmpty(overRideStatus)) {
       if (logger.isDebugEnabled())
           logger.debug("Overriding authStatus from action. action=" +
                       ruleAction + ", authSatus=" +
                       overRideStatus);
       newStatus = Integer.parseInt(overRideStatus);
   }

   String device = prefs.getCurrentDevice();
   int clientType = UIOUtil.CLIENT_TYPE_LOGIN;
   if (!StringUtil.isEmpty(device)){
     if (device.equals(Preferences.AUTH_DEVICE_TYPE_DEVICE_TEXTPAD)) {
         clientType = UIOUtil.CLIENT_TYPE_TEXTPAD;
     } else if (device.equals(Preferences.AUTH_DEVICE_TYPE_DEVICE_KEYPAD_FULL) ||
         device.equals(Preferences.AUTH_DEVICE_TYPE_DEVICE_KEYPAD_ALPHA)) {
         clientType = UIOUtil.CLIENT_TYPE_KEYPAD;
     } else if (device.equals(Preferences.AUTH_DEVICE_TYPE_DEVICE_HTML_CONTROL))
{
         clientType = UIOUtil.CLIENT_TYPE_NORMAL;
     } else if (device.equals(Preferences.AUTH_DEVICE_TYPE_DEVICE_QUESTIONPAD)) {
         clientType = UIOUtil.CLIENT_TYPE_QUESTIONPAD;
     } else if (device.equals(Preferences.AUTH_DEVICE_TYPE_DEVICE_PINPAD)) {
         clientType = UIOUtil.CLIENT_TYPE_PINPAD;
     }
   }

   sessionData.setClientType(clientType);

   if (newStatus != authStatus) {
     //Update tracker if status has changed
     sessionData.setAuthResult(newStatus);
     UIOUtil uioUtil = UIOUtil.instance();

     try {
       uioUtil.updateAuthStatus(sessionData, sessionData.getClientType(),
newStatus);
     } catch (BharosaProxyException bpe){
       logger.error("Failed to update auth status to " + newStatus + " while
processing ruleAction = " + ruleAction, bpe);
     }
   }
```

```
    return ruleAction;
  }

}
```

22.8 Integration Processors

You can use the integration processor to develop custom java code at `Entry`, `Exit`, and `Error` points in the OAAM user flow for each application in the OAAM environment. The Integration Processor also enables you to customize access to `AuthManager`, `PasswordManager`, and `UserDataManager` implementations.

This section contains the following topics:

- IntegrationProcessorIntf Interface
- Common User Flows
- Integration Processor Parameters

22.8.1 IntegrationProcessorIntf Interface

You can integrate your custom java code at `entry`, `exit`, and `error` points with client-side OAAM code.

```
public interface IntegrationProcessorIntf extends Serializable {

  public String onEntry();

  public String onExit(String target);

  public String onError(String target);

  public String getAppId();

  public FlowIntf getFlow();

  public AuthManagerIntf getAuthManager();

  public PasswordManagerIntf getPasswordManager();

  public UserDataManagerIntf getDataManager(String key);

}
```

22.8.2 Common User Flows

You can set up the `sessionData` object to begin common user flows. You can initiate the flows by executing the Start method.

Table 22–6 list the out-of-the-box objects and parameters:

*denotes optional parameter

Table 22–6 SessionData Objects

Class	Parameters	Notes
LoginFlow	String appId Map<String, String> rulesMap* Hashtable appContext*	Standard login flow with username page, password page. Fingerprinting, challenge and registration will occur as needed.
LoginNoUsernameFlow	String appId String loginId Map<String, String> rulesMap* Hashtable appContext*	Similar to the login flow, but will skip the username page and use a provided username instead.
AuthenticateFlow	String appId String loginId int authStatus Map<String, String> rulesMap* Hashtable appContext*	Similar to the login flow, but will skip the username and password pages using provided username and authentication status.
ForgotPasswordFlow	String appId String loginId Map<String, String> rulesMap* Hashtable appContext*	Begins the forgot password flow for the provided username. Fingerprinting and forgot password checkpoint rules will be executed, resulting in challenge if policies are configured to do so.
UserPreferencesFlow	String appId String loginId* int authStatus* Map<String, String> rulesMap* Hashtable appContext*	Begins the user preferences flow. If the user session already exists username and auth status will be used from sessionData. Alternatively the username and authentication status can be provided to create a user session and display user preferences.

22.8.3 Integration Processor Parameters

The following properties are used in the OAAM server framework for registering custom java classes.

22.8.3.1 Check for Integration ID

The request parameter/header to check for integration ID is

```
oaam.server.integration.param=intg
```

22.8.3.2 Integration Processor Registration

The integration processor allows for custom java class to handle OAAM server entry, exit, and error cases.

```
oaam.server.integration.processor.default=com.bharosa.uio.processor.integration.
DefaultIntegrationProcessor
oaam.server.integration.processor.oam=com.bharosa.uio.processor.integration.
OAMIntegrationProcessor
oaam.server.integration.processor.mobile=com.bharosa.uio.processor.integration.
MobileIntegrationProcessor
oaam.server.integration.processor.juniper=com.bharosa.uio.processor.integration.
JuniperSSLIntegrationProcessor
```

22.8.3.3 Oracle Access Management Access Manager Specific Integration Properties for Authentication Levels

The following are Oracle Access Management Access Manager-specific integration properties for authentication levels (used by the OAMIntegrationProcessor class):

```
oaam.server.integration.oamauthentication.level.enum=Enum for oam authentication
levels provided in tap token
oaam.server.integration.oamauthentication.level.enum.oamoaamlevelmapping=0
oaam.server.integration.oamauthentication.level.enum.oamoaamlevelmapping.name=
Mapping
oaam.server.integration.oamauthentication.level.enum.oamoaamlevelmapping.
oamauthlevels= -1 to 0, 1 to 99
oaam.server.integration.oamauthentication.level.enum.oamoaamlevelmapping.
oaamauthlevels= 0,1
```

22.9 Provider Registration

Providers allow for custom java classes to handle authentication and password management.

22.9.1 Authentication Manager

Authentication managers allow for custom integration with an external authentication service.

Custom implementations of authentication manager should extend the `com.bharosa.uio.manager.auth.AbstractAuthManager` class, which implements the `com.bharosa.uio.manager.auth.AuthManagerIntf` interface. Authentication manager returns a `com.bharosa.uio.manager.auth.AuthResult` object.

The provider can be registered here for any number of application IDs, or override the `getAuthManager` class in an Integration Processor.

To register your custom provider, set the following property:

```
bharosa.uio.default.password.auth.provider.classname =
 com.bharosa.uio.manager.auth.DummyAuthManager
```

`AbstractAuthManager` implements the `AuthManagerIntf` interface to perform user authentication.

Table 22–7 AbstractAuthManager

Method	Description
public AuthResult authenticate(VCryptAuthUser authUser, String password, Hashtable appContext) throws Exception	Performs internal logging and calls authenticateUser method
protected abstract AuthResult authenticateUser(VCryptAuthUser authUser, String password, Hashtable appContext) throws Exception	Abstract method signature that takes OAAM user object, password, and application context from UIOSessionData. Implementation of this method should call external authentication service to construct and return AuthResult object.

22.9.2 Password Manager

Password Manager classes allow for custom handling of password management. Custom password management implementations should implement the `com.bharosa.uio.manager.user.PasswordManagerIntf` interface.

The password manager allows for customization of change password, set password, and retrieval of password policy text that can be displayed to user to indicate password format requirements.

The provider can be registered here for any number of application IDs, or override the `getPasswordManager` class in an Integration Processor. OAAM's default implementation is only a placeholder and does not perform any actual password management.

To register your custom provider through properties, set the following:

```
bharosa.uio.default.user.management.provider.classname =
 com.bharosa.uio.manager.user.DefaultPasswordManager
```

`PasswordManagerIntf` is the interface for implementing password manager.

Table 22–8 PasswordManagerIntf

Method	Description
public VCryptResponse changePassword(UIOSessionData sessionData, String oldPasswd, String newPasswd, String confirmPasswd)	Allows for calling of external service that requires existing password to set new password. Used for password change.
public VCryptResponse setPassword(UIOSessionData sessionData, String password)	Allows for calling of external service that does not require existing password. Used for password reset such as in a forgot password flow.
public VCryptObjectResponse<String[]> getPasswordPolicyText(UIOSessionData sessionData)	Allows for custom password policy text to be displayed on password change/reset pages. This could be any messaging that the user should see, but is intended for notifying the user of password format and restrictions.

22.9.3 User Data Manager

User data managers allow customization of access and management of user profile data, such as mobile phone number and email address. By default, OAAM stores and retrieves user data from the OAAM database. With a custom implementation of the user data manager, an integration could call an external service.

When using an external service for user data, it is recommended to set a flag in the OAAM database that is used to track the existence of the user data being set.

To customize use data manager, it is recommended that you extend the `com.bharosa.uio.manager.user.DefaultContactInfoManager` class. This class implements the `com.bharosa.uio.manager.user.UserDataManagerIntf` interface.

Each user data field can use a different user data manager. This would allow each data element to be maintained in a different external service if required. By default OAAM uses the same data manager for all user data fields.

To register your custom user data manager, set the "`managerClass`" property on the "`userinfo.inputs`" enum.

Example:
`bharosa.uio.default.userinfo.inputs.enum.mobile.managerClass=com.bharosa.uio.manager.user.DefaultContactInfoManager`

`UserDataManagerIntf` is the user data manager interface for storing/retrieving user data.

Table 22–9 UserDataManagerIntf

Method	Description
public String getUserData(UIOSessionData sessionData, String key)	Get the user data value for the "key".
public void setUserData(UIOSessionData sessionData, String key, String value)	Set the user data "value" for the "key".

`DefaultContactInfoManager` - Default implementation of user data manager. This implementation splits the setting of the data value and the flag that indicates the value is set.

Table 22–10 DefaultContactInfoManager

Method	Description
public String getUserData(UIOSessionData sessionData, String key)	Calls getUserDataValue if getUserDataFlag returns true.
public void setUserData(UIOSessionData sessionData, String key, String value)	Calls setUserDataValue and setUserDataFlag for the "key" and "value".
protected String getUserDataValue(UIOSessionData sessionData, String key)	Gets the "value" for the "key" from the OAAM database.
protected void setUserDataValue(UIOSessionData sessionData, String key, String value)	Sets the "value" for the "key" in the OAAM database.
protected boolean getUserDataFlag(UIOSessionData sessionData, String key)	Returns true if the flag for the give "key" is set in the OAAM database.
protected void setUserDataFlag(UIOSessionData sessionData, String key, String value)	Sets the flag indicating if a value is set for a user data key. If value passed is empty, the flag is cleared.

22.10 Legacy Rules Result Processors

The legacy rule result processors support 10g policies. The properties are listed as follows:

```
bharosa.uio.default.rules.result.processor.PasswordTextPadGeneric =
 com.bharosa.uio.processor.rules.result.legacy.PasswordRulesResultProcessor
bharosa.uio.default.rules.result.processor.PasswordTextPad =
 com.bharosa.uio.processor.rules.result.legacy.PasswordRulesResultProcessor
bharosa.uio.default.rules.result.processor.PasswordKeypad =
 com.bharosa.uio.processor.rules.result.legacy.PasswordRulesResultProcessor
bharosa.uio.default.rules.result.processor.PasswordKeypadFull =
 com.bharosa.uio.processor.rules.result.legacy.PasswordRulesResultProcessor
bharosa.uio.default.rules.result.processor.PasswordHTML =
 com.bharosa.uio.processor.rules.result.legacy.PasswordRulesResultProcessor

bharosa.uio.default.rules.result.processor.RegisterUserOptionalQuestionPad =
 com.bharosa.uio.processor.rules.result.legacy.RegisterRulesResultProcessor
bharosa.uio.default.rules.result.processor.RegisterUserQuestionPad =
 com.bharosa.uio.processor.rules.result.legacy.RegisterRulesResultProcessor
bharosa.uio.default.rules.result.processor.RegisterUserOptionalTextPad =
 com.bharosa.uio.processor.rules.result.legacy.RegisterRulesResultProcessor
bharosa.uio.default.rules.result.processor.RegisterUserTextPad =
 com.bharosa.uio.processor.rules.result.legacy.RegisterRulesResultProcessor
bharosa.uio.default.rules.result.processor.RegisterUser =
 com.bharosa.uio.processor.rules.result.legacy.RegisterRulesResultProcessor
bharosa.uio.default.rules.result.processor.RegisterUserOptional =
```

```
 com.bharosa.uio.processor.rules.result.legacy.RegisterRulesResultProcessor
bharosa.uio.default.rules.result.processor.RegisterQuestionsQuestionPad =
 com.bharosa.uio.processor.rules.result.legacy.RegisterRulesResultProcessor
bharosa.uio.default.rules.result.processor.RegisterQuestionsTextPad =
 com.bharosa.uio.processor.rules.result.legacy.RegisterRulesResultProcessor
bharosa.uio.default.rules.result.processor.RegisterQuestions =
 com.bharosa.uio.processor.rules.result.legacy.RegisterRulesResultProcessor
bharosa.uio.default.rules.result.processor.RegisterImageTextPad =
 com.bharosa.uio.processor.rules.result.legacy.RegisterRulesResultProcessor
bharosa.uio.default.rules.result.processor.RegisterImageKeyPad =
 com.bharosa.uio.processor.rules.result.legacy.RegisterRulesResultProcessor

bharosa.uio.default.rules.result.processor.ChallengeQuestionPad =
 com.bharosa.uio.processor.rules.result.legacy.ChallengeRulesResultProcessor
bharosa.uio.default.rules.result.processor.ChallengeSMS =
 com.bharosa.uio.processor.rules.result.legacy.ChallengeRulesResultProcessor
bharosa.uio.default.rules.result.processor.ChallengeEmail =
 com.bharosa.uio.processor.rules.result.legacy.ChallengeRulesResultProcessor
bharosa.uio.default.rules.result.processor.ChallengeIM =
 com.bharosa.uio.processor.rules.result.legacy.ChallengeRulesResultProcessor
bharosa.uio.default.rules.result.processor.ChallengeVoice =
 com.bharosa.uio.processor.rules.result.legacy.ChallengeRulesResultProcessor
bharosa.uio.default.rules.result.processor.ChallengeQuestion =
 com.bharosa.uio.processor.rules.result.legacy.ChallengeRulesResultProcessor
```

23

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 chapter contains the following sections:

This document assumes that you are familiar with the concepts of OAAM Offline.

23.1 Developing a Custom Loader for OAAM Offline

The abstract classes for the custom loader are in `oaam_core.jar`, which is located in the `oaam/cli/lib` folder in the *`IDM_Home`* directory.

To deploy your custom loader, follow these steps:

1. Extract the `oracle.oaam.extensions.war` file under the *`ORACLE_MW_HOME`*`/Oracle_IDM1/oaam/oaam_extensions/generic` folder.
2. Place your jar file into `WEB-INF/lib` folder.
3. Repackage the `oracle.oaam.extensions.war` file.
4. From the Oracle WebLogic Administration Console, update and restart the `oracle.oaam.extensions` library, and restart the OAAM Offline application.

   The custom loader loads transactions from an OAAM server database.

23.2 Base Framework

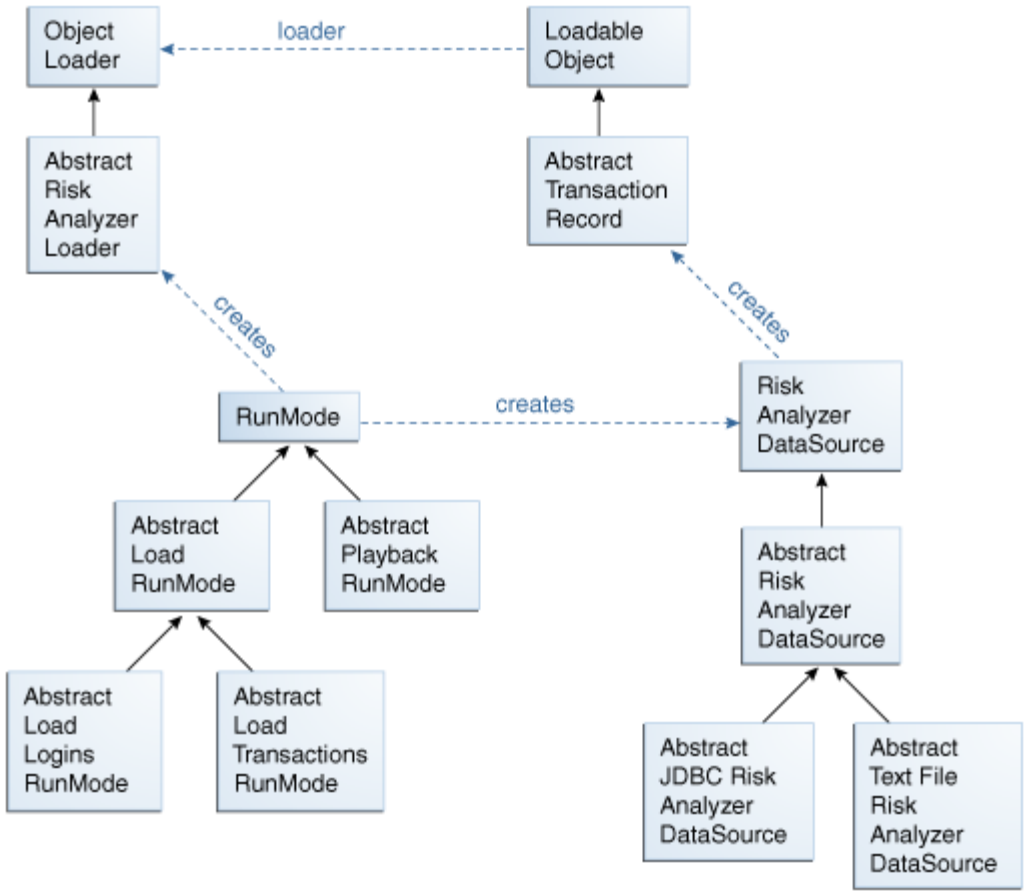
A custom loader is required only if the data from sources other than a database, data other than login, or complex data is needed for the OAAM Offline task.

23.2.1 Overview

The OAAM Offline custom loader consists of the following key parts:

- loadable object
- data source
- loader
- run modes

Figure 23–1 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.

23.2.2 Important Classes

Table 23–1 provides a summary of the different data loader classes.

Table 23–1 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 verify 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.

23.2.3 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();
}
```

23.3 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.

23.3.1 Default Load Implementation

The default load implementation is summarized in Figure 23–2.

Figure 23–2 Default Load Implementation

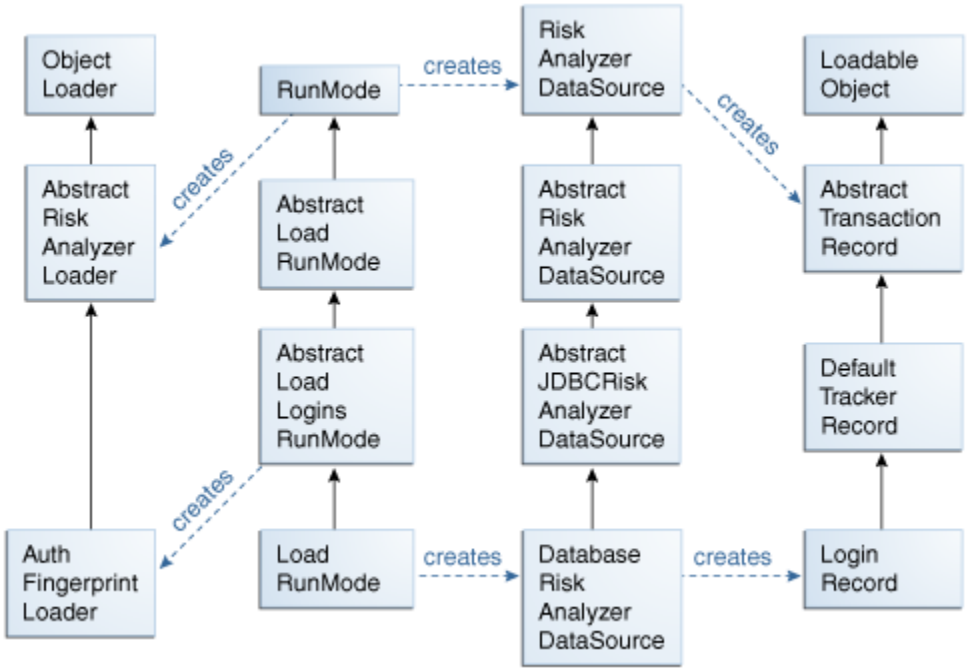


Table 23–2 Default Implementation

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.

23.3.2 Default Playback Implementation

The default playback implementation is summarized in Figure 23–3.

Figure 23–3 Default Playback Implementation

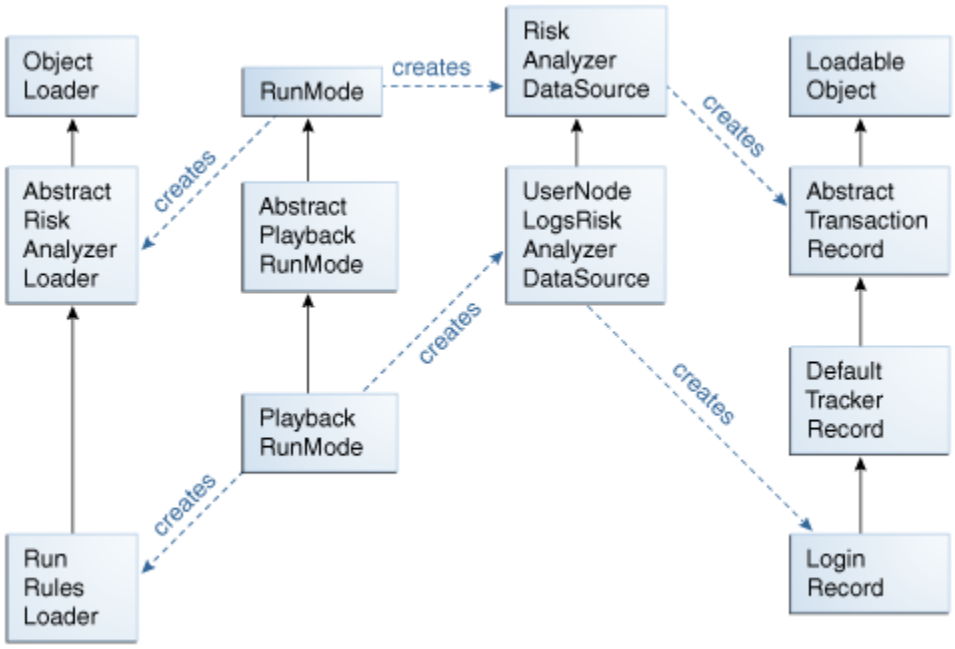


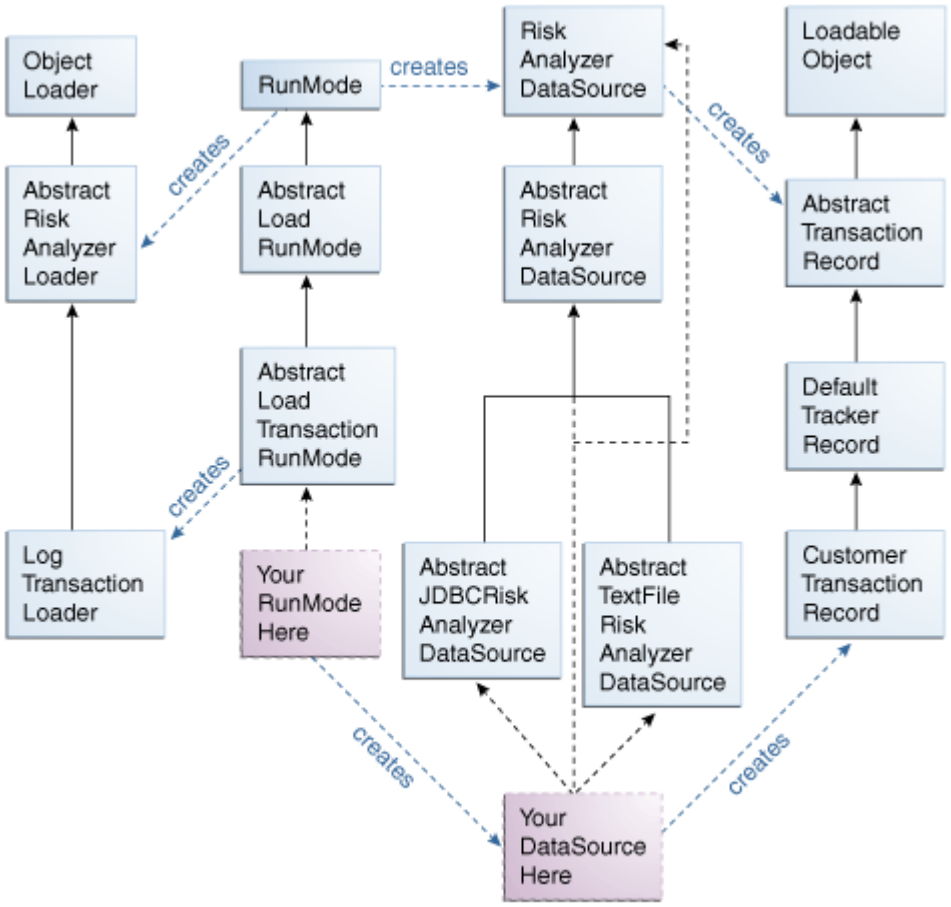
Table 23–3 Default Playback Implementation

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.

23.4 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 23–4 Overriding the Loader or Playback Behavior



23.5 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 a couple of ways to do this: extending `AbstractJDBCRiskAnalyzerDataSource` or extending `AbstractRiskAnalyzerDataSource`.

23.5.1 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 must 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.

23.5.2 Extending AbstractRiskAnalyzerDataSource

If `AbstractJDBCRiskAnalyzerDataSource` is not appropriate, then you will need to extend `AbstractRiskAnalyzerDataSource` instead. For example, 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 is necessary 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.

Loading from a Text File

If a file based custom loader has to be used, extend the AbstractRiskAnalyzerDataSource and implement the custom class by seeing what AbstractTextFileRiskAnalyzerDataSource does and copying the code from AbstractTextFileRiskAnalyzerDataSource.

23.6 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 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 must start from when the job is resumed.

23.6.1 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.

23.6.2 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.

23.6.3 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.

24

Creating OAAM Oracle BI Publisher Reports

This chapter contains instructions on creating Oracle BI Publisher reports on data in the OAAM database schema.

This chapter contains the following sections:

- Create Oracle BI Publisher Reports on Data in the OAAM Database Schema
- Building OAAM Transactions Reports

24.1 Create Oracle BI Publisher Reports on Data in the OAAM Database 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.

24.1.1 Create a Data Model

For instructions on creating a new report, see *Oracle Business Intelligence Publisher Report Designer's Guide*.

24.1.2 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. For more information about OAAM User Defined Enums, see Chapter 7, "Using the OAAM Extensions Shared Library to Customize OAAM." 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 must display English only or you must display internationalized strings.

24.1.2.1 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.

24.1.2.2 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 must add *`alias`*.label to the select clause.

24.1.2.3 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 must add *`alias`*.config_value to the select clause.

24.1.3 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.

24.1.3.1 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.

24.1.3.2 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.

24.1.4 Adding Geolocation Data

The OAAM database 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 typically 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))
```

24.1.5 Adding Sessions and Alerts

Sessions and alerts exist in the VCRYPT_TRACKER_USERNODE_LOGS and VCRYPT_ALERT tables, respectively. They join to each other through the REQUEST_ID field, and they each join to the geolocation data through the VCRYPT_IP_LOCATION_MAP table through the BASE_IP_ADDR field.

24.1.5.1 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.

Table 24–1 lists the type code fields and the names of the user defined enum in VCRYPT_TRACKER_USERNODE_LOGS.

Table 24–1 VCRYPT_TRACKER_USERNODE_LOGS

Field Name	User Defined Enum Name
AUTH_STATUS	auth.status.enum
AUTH_CLIENT_TYPE_CODE	auth.client.type.enum

Table 24–2 lists the type code fields and the name of the user defined enums in VCRYPT_ALERT.

Table 24–2 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

24.1.6 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
```

24.1.7 Adding Layouts to the Report Definition

BI Publisher offers several options for designing templates for your reports. For instructions on designing templates, see *Oracle Business Intelligence Publisher Report Designer's Guide*.

24.2 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

24.2.1 Get Entities and Transactions Information

To get the Transaction Definition key and Entity Definition keys, follow these steps:

1. Log in to OAAM Admin and navigate to the Transactions menu and search for the transaction definitions you are interested in.
2. Go to the **General** tab and write 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 write 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.

24.2.2 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.

24.2.2.1 Information about Data Types

For your reference, number data types are listed in Table 24–3.

Table 24–3 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

24.2.2.2 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;
```

24.2.2.3 Build Entity Data SQL Queries and Views

The preceding 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 preceding 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.

```
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>
```

24.2.3 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.

24.2.3.1 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;
```

24.2.3.2 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. Information on creating a view for entities and transactions is provided in Section 24.2.5, "Generating a Database View of Entities and Transactions."

```
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>
```

24.2.4 Build Reports

Follow the instructions in this section to build reports for entities and transactions.

24.2.4.1 Building Entity Data Reports

Use the SQL Queries or Views built using the information mentioned in Section 24.2.2.3, "Build Entity Data SQL Queries and Views."

24.2.4.2 Building Transaction Data Reports

Use the SQL Queries or Views built using the information mentioned in Section 24.2.3.2, "Build Transaction Data SQL Queries and Views."

24.2.4.3 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>`.

24.2.5 Generating a Database View of Entities and Transactions

OAAM persists entity and transaction data in the database. OAAM provides a command line tool to generate the SQL script file which contains SQL statements to create views for entities and transactions in OAAM.

These views help you view the transaction and entity related data in the database in an easier way as compared to querying specific tables for every detail since they provide a comprehensive picture of the entities and transactions currently available in the database along with information about their relationships (transaction-entity, entity-entity).

24.2.5.1 Generating the SQL Script File

The `generateTrxEntityViewsSQL` script creates a SQL file which upon execution create database views for existing transaction and entity related definitions and data in the database.

The script generates a SQL script on the basis of transaction and entity definitions present in the database.

24.2.5.1.1 Pre-requisites Before running the script, ensure the OAAM CLI environment is set up. For instructions on setting up the CLI environment, see *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*.

- The following must be added to the CSF/Credential Store using Oracle Enterprise Manager Fusion Middleware Control (<host>:<port>/em).

  OAAM database User Name and Password with oaam_db_key as the keyname under the map oaam.

- Table 24–4 shows the properties that must be set in the `oaam_cli.properties` file before you can generate the view.

Table 24–4 Properties to Set Before Running generateTrxEntityViewSQL

Property	Value
oaam.db.url	JDBC URL for the OAAM metadata repository
oaam.trxentityview.filename	Filename to store the generated SQL script. Default value is createTrxEntityViews.sql

24.2.5.1.2 Generate the SQL Script To generate the SQL script file, run the `generateTrxEntityViewsSQL` script from the OAAM CLI folder.

The default file generated is createTrxEntityViews.sql and contains some create or update queries to create or update views for each transaction and entity definition.

24.2.5.2 Creating the Database Views for Entities and Transactions

Follow the subsequent steps to create database views for entities and transactions stored in the OAAM database.

1. Log in to the database as a OAAM database schema user.
2. Grant the OAAM database schema user the Grant Create View privilege.
3. Connect to the database using the OAAM database schema user.

   For example, sqlplus *DEV_OAAM*/*PASSWORD*
4. Run `createTrxEntityViews.sql`.

   The script creates database views of each entity and transaction defined in the OAAM database.

24.2.5.3 Entity View Details

For each entity defined in OAAM, one view will be created with the name oaam_ent_*entity_key*. The *entity_key* will be replaced by the key of the entity as defined in OAAM. The created view will contain one column for each data defined in the entity. In addition to the data columns, the created view will contain the following columns:

- ENTITY_ID: Unique identifier of the entity instance
- CREATE_TIME: Time the entity was created
- UPDATE_TIME: Time of last update of the entity

24.2.5.4 Transaction View Details

For each transaction defined in OAAM, one view will be created with the name `oaam_trx_`*transaction_key*. The *transaction_key* will be replaced by the key of the transaction as defined in OAAM. The created view will contain one column for each data defined in the transaction.

The created view will contain one column for each entity referenced in the transaction to store the entity_id of the referenced entity. For example, the `entity_id` column in the `oaam_ent_`*entity_key* view. Spaces in the instance names will be replaced with an underscore in the column names.

- `LOG_ID`: Unique identifier for the transaction
- `USER_ID`: User who performed the transaction
- `REQUEST_ID`: Session in which this transaction was performed
- `EX_TRX_ID`: External ID of this transaction
- `STATUS`: Status of the transaction
- `CREATE_TIME`: Time the entity was created
- `UPDATE_TIME`: Time of last update of the entity
- `CREATED_HOUR`: Create time truncated to the nearest hour
- `CREATED_DAY`: Create time truncated to the nearest day

- CREATED_WEEK: Create time truncated to the nearest week
- CREATED_MONTH: Create time truncated to the nearest month
- CREATED_YEAR: Create time truncated to the nearest year

24.2.5.5 Identifiers

Oracle database limits the length of identifiers such as table, view and column names to 30 characters. To ensure that the views created by the script comply for this requirement, limit the name of entity, transaction, and datafield names to the following identifiers:

- entity: 21 (view names will be "oaam_ent_" + <entity_key>)
- transaction: 21 (view names will be "oaam_trx_" + <transaction_key>)
- datafield: 28 (column name will be "d_" + <data_element_name>)
- entity-ref: 20 (column name will be <relationship_name> + "_entity_id")

If the script finds any names longer than the above limits, the script will trim the identifier. Look for such trimmed column/view names while writing SQL queries on the created views.

Space, dash ("-") and period (".") characters in the names will be replaced with an underscore.

24.2.5.6 Example of SQL Query to Create a View

A typical SQL query to create a view based on an Entity definition "Address" is shown as follows. oaam_ent_ADDRESS is the view that is created from the SQL query.

```
create or replace
force view oaam_ent_ADDRESS
as
    (select
          (ent.ENTITY_ID, ent.EXT_ENTITY_ID, ent.ENTITYNAME, ent.ENTITY_KEY,
           ent.ENTITY_TYPE,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 row0 on
           (row0.ENTITY_ID = ent.ENTITY_ID
            and row0.ROW_ORDER = 0
            and row0.EXPIRE_TIME is null)
where entDef.ENTITY_DEF_KEY='Address'
and ent.ENTITY_DEF_ID = entDef.ENTITY_DEF_ID);
```

The tables used for the view are listed in Table 24–5.

Table 24–5 Entity Tables in the Entity View

Table	Description
VT_ENTITY_DEF	This table has definitions of all the different Entities.
VT_ENTITY_ONE	This table has the Entity Key, name, a unique ID and expiry.
VT_ENTITY_ONE_PROFILE	This table has the Entity data stored in it.

The view provides to you a single view which contains the entire information about the particular transaction/entity definition and all the data associated with that

definition. Once the script is run, you can choose to see the data in the view whenever needed. This view provides to you a consolidated view of data in one place instead of mapping data from various tables. The `oaam_ent_ADDRESS` view is shown in Table 24–6.

Table 24–6 oaam_ent_ADDRESS

Field Name	DB Type	Description
ENTITY_ID	BIGINT	ID of the entity
EXT_ENTITY_ID	VARCHAR	External entity ID (supplied by client)
ENTITYNAME	TEXT	Name of the entity (generated name of the entity by the namegen Scheme according to the entity definition).
ENTITY_KEY	TEXT	Key for the entity (generated key for the entity by the keygen scheme according to the entity definition). This key is used to perform the lookup whether this entity exists in the DB.
ENTITY_TYPE	INT	Type of the entity
CREATE_TIME	DATETIME	Date/Time when this object was created
UPDATE_TIME	TIMESTAMP	Date value
EXPIRY_TIME	DATETIME	Expiry date value. Set according to preconfigured property, number of days from the request time.
RENEW_TIME	DATETIME	Renew date value. After this time, if this entity is being used, the expiry is extended as well as the renew_time.
ROW_ORDER	INT	Row order (starts with 0 to accommodate any number of data. Once 10 columns are exhausted, another record with row_order 1 would be inserted and so on.)
EXPIRE_TIME	DATETIME	Date/time when this profile expires
ENTITY_DEF_KEY	TEXT	Key of the entity. For example, address, merchant, and so on.
ENTITY_DEF_ID	BIGINT	ID for the entity definition

25

Developing 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.

This chapter contains the following sections:

25.1 Adding a New Configurable Action

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. Ensure that the size of the parameters array returned is the same as the number of parameters.
   - `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 JAR files 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, see Section 25.4, "Sample JUnit Code."
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 file. For instructions for adding the custom JAR file to Oracle Adaptive Access Manager, see Section 7, "Using the OAAM Extensions Shared Library to Customize OAAM."
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 information on configuring Configurable Actions, see *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*.

25.2 Executing Configurable Actions in a Particular Order and Data Sharing

You can use configurable actions 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 implementor'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`.

25.3 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. In addition, check for a log entry or a database entry created by the Configurable Action.

25.4 Sample JUnit Code

The following is a 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 {

    }
}
```

25.5 Sample Java Code for Configuration Action

Sample code is provided in this section for a configurable action:

```
public class HelloWorldAction implements DynamicAction {

     private UserDefEnum valueTypeEnum = UserDefEnum.getEnum("value.type.enum");

     public boolean execute(String sessionId, Date requestTime,

               Map ruleContextMap, VCryptRulesResult ruleResult,
```

```
            Map actionParamValueMap, Map actionContextMap) throws Exception {
        // TODO Auto-generated method stub
         System.out.println("Hello World!!");
         return false;
 }

public DynamicActionParamInfo[] getParameters() {
    DynamicActionParamInfo params[] = new DynamicActionParamInfo[3];

    String paramName = "Sample Integer Parameter";
    String description = "Integer Parameter Description";
    String notes = "Integer Parameter Notes";
    String promptLabel = "Integer Parameter";
    int valueType = valueTypeEnum.getElementValue("int");
    String defaultValue = "1";

    params[0] = new DynamicActionParamInfo();

    params[0].setParamName(paramName);
    params[0].setPromptLabel(promptLabel);
    params[0].setNotes(notes);
    params[0].setDescription(description);
    params[0].setValueType(valueType);
    params[0].setDefaultValue(defaultValue);

    paramName = "Sample String Parameter";
    description = "String Parameter Description";
    notes = "String Parameter Notes";
    promptLabel = "String Parameter";
    valueType = valueTypeEnum.getElementValue("string");
    defaultValue = "Sample String value";
```

```
        params[1] = new DynamicActionParamInfo();

        params[1].setParamName(paramName);

        params[1].setPromptLabel(promptLabel);

       params[1].setNotes(notes);

       params[1].setDescription(description);

       params[1].setValueType(valueType);

       params[1].setDefaultValue(defaultValue);



       paramName = "Sample Boolean Parameter";

       description = "Boolean Parameter Description";

       notes = "Boolean Parameter Notes";

       promptLabel = "Boolean Parameter";

       valueType = valueTypeEnum.getElementValue("boolean");

       defaultValue = "true";

       params[2] = new DynamicActionParamInfo();

       params[2].setParamName(paramName);

       params[2].setPromptLabel(promptLabel);

       params[2].setNotes(notes);

       params[2].setDescription(description);

       params[2].setValueType(valueType);

       params[2].setDefaultValue(defaultValue);
                              return params;

    }

}
```

26

Creating Checkpoints and Final Actions

A checkpoint is a specified point in a session when Oracle Adaptive Access Manager collects and evaluates security data using the rules engine.

New checkpoints can be added and existing checkpoint properties can be modified using the Properties Editor.

This chapter provides information on how to create and configure a new checkpoint and how to modify an existing checkpoint. It includes the following sections:

26.1 Creating a New Checkpoint

To create a checkpoint, use the Properties Editor.

The following checkpoint enumeration is shown for your reference.

```
profile.type.enum.nameofcheckpoint=Checkpoint_Value
profile.type.enum.nameofcheckpoint.name=Checkpoint_Name
profile.type.enum.nameofcheckpoint.description=Checkpoint_Description
profile.type.enum.nameofcheckpoint.ruleTypes=user,device,location
profile.type.enum.nameofcheckpoint.listTypes=vtusers
profile.type.enum.nameofcheckpoint.finalactionrule=process_results.rule
profile.type.enum.nameofcheckpoint.isPreAuth=true
```

The Checkpoint value must unique number. Make sure no other checkpoint uses the identifier. This ID is like a primary key in database terminology. For example, "1001."

The Checkpoint name must be user-presentable and meaningful. The name is used in Oracle Adaptive Access Manager.

If the checkpoint creation is successful, add the appropriate properties by clicking the **Add New** button under the Properties box.

The Checkpoint's required properties are:

- finalactionrule=process_results.rule

  The "finalactionrule" property specifies the Rule file that decides the final action. When the Rules Engine processes the policies for the checkpoint, it determines the score and a list of actions. The final action list is list of action that are deemed final when rules are run. The rule file is consulted to see what action should be given as

final action. If you are not sure, set the value as in the other checkpoints.The out-of-the-box "process_results.rule" file is sufficient for most actions.

An example of the rule file is provided in Section 26.4, "Final Action."

- listTypes= vtusers

  Always set listTypes to "vtusers."

  The policy can be linked to only usergroups.

- ruleTypes= user,device,location,in_session

  The "ruleTypes" property defines the list of rule types supported during the checkpoint. Depending on the context of the checkpoint, possible values are "user," "device," "location," and "in_session." Use commas to separate multiple values. You can use all rules of the comma separated types in this checkpoint.

  For example if you set ruleTypes to "user,location," you can use the rules of the type "user" and "location" in the checkpoint, and the user and location information is available for this checkpoint.

  Another example, for the "Cancel Order" checkpoint, if "user,device,location" are specified for ruleTypes, the "user" Rule type expects that the user information to be available during the "Cancel Order" checkpoint. If the user information is not available at the time of the "Cancel Order" checkpoint, "user" should not be included in the list.

Other properties you may add are:

- isPreAuth

  True indicates that this checkpoint is a pre-authentication checkpoint. OAAM Admin updates the user details with the pre-auth score and pre-auth action. The default for isPreAuth is "false." There cannot be two checkpoints with this flag set to "true." Also the same checkpoint cannot be marked as postAuth and preAuth.

- isPostAuth

  True indicates that this checkpoint is a post-authentication checkpoint. OAAM Admin updates the user details with the post-auth score and post-auth action. The default for isPostAuth is "false." There cannot be two checkpoints with this flag set to "true." Also the same checkpoint cannot be marked as postAuth and preAuth.

After creating the checkpoint, you must restart the server.

26.2 Creating a Checkpoint Example

An example for creating the "addressChange" checkpoint is shown:

```
profile.type.enum.addressChange=88
profile.type.enum.addressChange.name=Address Change
profile.type.enum.addressChange.description=Address Change checkpoint
profile.type.enum.addressChange.ruleTypes=user,device,location
profile.type.enum.addressChange.listTypes=vtusers
profile.type.enum.addressChange.finalactionrule=process_results.rule
profile.type.enum.addressChange.isPreAuth=true
```

For finalactionrule, "process_results.rule" was provided because the Final Action for a given checkpoint during rules evaluation is determined by this rule file. File process_results.rule is supplied out-of-the-box and no additional steps are required.

26.3 New Action

A new action can be defined through OAAM Admin when you define action groups and when you want to add an action, which gives a choice to create an action. You can also create an action by adding an element to `rule.action.enum`.

26.4 Final Action

If a new action is defined and it is the final action of a checkpoint, you must perform the following steps.

1. Locate the `rule` file for that checkpoint or define one if you are creating a new checkpoint. This file ensures that the action that is supposed to be final is available in the final action list when the rules are run.

   If you do not want to use the default file, the subsequent example is provided to illustrate what the file looks like. You will want to include the rule file in the OAAM Extensions Library which is in the OAAM Server's classpath.

```
==
<rule name="Block" no-loop="true" salience="100">
    <parameter identifier="actionList">
        <class>java.util.List</class>
     </parameter>
.
     <java:condition>actionList.contains("Block")</java:condition>
.
     <java:consequence>
        if (logger != null){
        logger.debug("Executing Block condition");
         }
 .
        finalAction.append("Block");
        drools.clearAgenda();  <!-- This stops any other rules from being
evaluated -->
      </java:consequence>
</rule>
```

2. Locate or create the `finalactionrule` property for that checkpoint.

   For example, for checkpoint "X" it will be

```
profile.type.enum.X.finalactionrule=some_file_name.rule
```

3. Add the rule file to a shared library and make that library available to the OAAM Server. You can package the file in the OAAM_extensions library since it is available to server.

The previous example shows that the final action list contains the `Block` action. If you are defining your own checkpoint, you may also want to perform the previous steps and set your `finalactionrule` property to point to the file created. The preceding `rule` file must be in the classpath of the servers so that oaam_server and oaam_offline server can use this information. After defining rule file or creating such file, the servers will have to be restarted.

Part VI

Lifecycle Management

Part VI contains the following chapters:

27

Migrating Native Applications to OAAM 11*g*

This chapter covers the tasks involved in migrating an existing natively integrated 10.1.4.5 application to 11*g*.

This chapter contains the following sections:

27.1 Prerequisites for Migration of an Existing Natively Integrated 10.1.4.5 Application

You must follow these prerequisites for migrating your existing natively integrated application:

- Client must use the OAAM Shared Library for Native Integration which uses SOAP.
- Client must specify the configurable properties in `oaam_custom.properties` and this file must be in the Java Classpath of the client application.
- See Section 27.4, "Migrating Native Applications that Cannot Use OAAM Shared Library" if the Native Application cannot use the OAAM Shared Library

27.2 Migrating Native Static Linked (In-Proc) Applications to OAAM 11*g*

To migrate the natively integrated in-proc application to OAAM 11*g*, you must:

- Add a reference to the OAAM Shared Library in the `weblogic.xml` file so that you can use the OAAM Shared Library.
- Move all configurable properties to the custom properties file.

27.2.1 Use the OAAM Shared Library Instead of Static Linking to OAAM JAR Files

To use the OAAM Shared Library in Web applications, you must reference the `oracle.oaam.libs` shared library in the WebLogic deployment descriptor file, `weblogic.xml` file. Add the following entry to `weblogic.xml`:

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

27.2.2 Move All Configurable Properties into the oaam_custom.properties File

As part of migrating the application, you must perform these steps:

1. Move all the configurable properties to `oaam_custom.properties`.

   In 10g all custom configuration overrides were created in the `bharosa_client.properties` file.

2. Remove/delete all other OAAM property files from the native application.
3. Remove/delete all old OAAM JAR files.

27.3 Migrating Native SOAP Applications to OAAM 11*g*

Follow the procedures in this section to migrate your native SOAP application to OAAM 11*g*.

27.3.1 Use OAAM Shared Library Instead of Static Linking to OAAM JAR Files

To use the OAAM Shared Library in Web applications, you must reference the `oracle.oaam.libs` shared library in the WebLogic deployment descriptor file, `weblogic.xml` file. Add the following entry to `weblogic.xml`:

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

27.3.2 Move All Configurable Properties into the oaam_custom.properties File

As part of migrating the application, you must perform these steps:

1. Move all the configurable properties to `oaam_custom.properties`.
2. Add the following properties to `oaam_custom.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.

3. Remove/delete all other OAAM property files from the native application.
4. Remove/delete all old OAAM JAR files.

27.3.3 Configure SOAP/WebServices Access

For details on configuring SOAP/WebServices Access, refer to Section 4.3, "OAAM SOAP Integration.".

27.4 Migrating Native Applications that Cannot Use OAAM Shared Library

The following process covers migrating your existing 10.1.4.5 Natively Integrated application that is currently using SOAP authentication to 11*g*.

27.4.1 Use the OAAM 11*g* JAR Files

After those files are copied, you can copy the `oaam_core.jar` file from the *`$ORACLE_HOME`*`/oaam/cli/lib` folder into your applications library folder. *`$ORACLE_HOME`* is usually the `ORACLE_IDM1` folder in the Middleware Home.

27.4.2 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` folder.

27.4.3 Specify the Configurable Properties in the oaam_custom.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 `oaam_custom.properties` file. This was typically the file modified on the server side for the same purpose. A `oaam_custom.properties` file that contains the contents of your old `bharosa_client.properties` with the addition of the following new properties must 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 set up in 10*g*, you will need to refer to "Setting Up Encryption" in the *Oracle Adaptive Access Manager Installation and Configuration Guide*, Release 10g (10.1.4.5) for creating a SOAP keystore for use with the new 11*g* environment.

28

Handling Lifecycle Management Changes

Because of integrated deployment of Oracle Adaptive Access Manager with other applications, Oracle Virtual Directory, Oracle Identity Manager, Oracle Access Management 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 these sections:

- Oracle Virtual Directory (OVD) Host, Port, and SSL Enablement Changes
- Oracle Identity Manager (OIM) URL Changes
- Oracle Access Management Access Manager Host and Port Changes
- Oracle Internet Directory (OID) Host and Port Changes and SSL Enablement
- Database Host and Port Changes
- Moving Oracle Adaptive Access Manager to a New Production Environment
- Moving Oracle Adaptive Access Manager to an Existing Production Environment

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

28.1 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. Navigate to OAAM Admin:

   `http://`*`OAAM_Managed_Server_Host`*`:`*`OAAM_Admin_Managed_Server_Port`*`/oaam_admin`

3. Log in as a user with access to the OAAM Properties Editor.
4. Open the OAAM Property Editor modify parameters:
   - Change the password authentication provider to LDAP.
   - Rewire existing Oracle Adaptive Access Manager for Oracle Virtual Directory host name.
   - 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 28–1 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 *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*.

5. Restart the Oracle Adaptive Access Manager server-related managed server.

28.2 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. Navigate to OAAM Admin:

   `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 28–2 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 Management 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 Management Access Manager and Oracle Identity Manager.

For information on setting properties in Oracle Adaptive Access Manager, see *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*.

5. Restart the Oracle Adaptive Access Manager server-related managed server.

28.3 Oracle Access Management Access Manager Host and Port Changes

For information on rewiring Oracle Access Management Access Manager for Oracle Adaptive Access Manager host name and port changes, see *Oracle Fusion Middleware Administrator's Guide for Oracle Access Management*.

28.4 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. Navigate to OAAM Admin:

   `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 host name
- 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 28–3 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 *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*.

5. Restart the Oracle Adaptive Access Manager server-related managed server.

28.5 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. Navigate 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 Oracle WebLogic Administration Console

To changes the data source:

1. In the Oracle WebLogic Administrative Console, navigate to **Services**, select **JDBC**, select **Data Sources**, and then **oaamDS**.
2. Click **oaamDS** and edit it for host name/port or user name/password.

28.6 Moving Oracle Adaptive Access Manager to a New Production Environment

For information on moving Oracle Adaptive Access Manager to a new production environment, see *Oracle Fusion Middleware Administrator's Guide*.

28.7 Moving Oracle Adaptive Access Manager to an Existing Production Environment

For information on moving Oracle Adaptive Access Manager to an existing production environment, see *Oracle Fusion Middleware Administrator's Guide*.

Part VII

Troubleshooting

Part VII contains the following chapter:

- Chapter 29, "FAQ/Troubleshooting"

29

FAQ/Troubleshooting

This chapter provides troubleshooting tips and answers to frequently asked questions. It contains the following sections:

29.1 Using My Oracle Support for Additional Troubleshooting Information

You can use My Oracle Support (formerly MetaLink) to help resolve Oracle Fusion Middleware problems. My Oracle Support contains several useful troubleshooting resources, such as:

- Knowledge base articles
- Community forums and discussions
- Patches and upgrades
- Certification information

> **Note:** You can also use My Oracle Support to log a service request.

You can access My Oracle Support at `https://support.oracle.com`.

29.2 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:

29.2.1 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.
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 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 occurs only at a certain site, then compare the differences.
Divide	Break the problem down

29.2.2 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).

Process	Description
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.

29.2.3 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

29.2.4 Process Flow of Analysis

The process flow of analysis is presented as follows:

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.

29.2.4.1 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?

Step	Description
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.

29.2.4.2 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.
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.

29.2.4.3 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.

Considerations	Description
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

29.2.4.4 IS and IS NOT but COULD BE

Consider what the problem is, what it is not, 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
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.

29.2.4.5 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

29.2.4.6 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

29.2.4.7 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 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, that is, 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.

29.2.4.8 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

29.3 Troubleshooting Tools

This section contains information about tools and processes you can use to investigate and troubleshoot issues with your system.

Table 29–1 lists the general and OAAM-specific tools you can use for troubleshooting problems.

Table 29–1 Troubleshooting Tools

Category	Description
General Tools	■ Oracle Enterprise Manager Fusion Middleware Control ■ Database Enterprise Manager ■ Monitor Data in DMS ■ Audit Data ■ Ping/Network Check Tools
OAAM Specific Tools	■ Dashboard ■ Monitor Data ■ Log files

Table 29–2 provides items to check for when troubleshooting the system.

Table 29–2 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 Oracle Enterprise Manager Fusion Middleware Control	Make sure the log level is changed appropriately before tracing and debugging
Search for log messages through Oracle Enterprise Manager Fusion Middleware Control	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 used to correlate individual events as being part of the same request execution flow.
Use the Oracle WebLogic Administration Console to monitor database connection pool	Check the health of the connection pool through the Oracle WebLogic Administration Console.

Table 29–3 summarizes problems and the checks you can perform to troubleshoot and solve the problem.

Table 29–3 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 11.1.2 ■ 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 parameters if necessary ■ Check the database using Oracle Enterprise Manager Fusion Middleware Control and see if there are any queries that are slow. Follow recommendation for adding suggested indexes in *Oracle Fusion Middleware Performance and Tuning Guide*. ■ 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's WebLogic Server 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 Oracle WebLogic Administration Console and make sure all web applications are targeted properly to their managed servers

Table 29–3 (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 ■ Oracle Web Services Manager (OWSM) is enabled by default, so you must set the 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 file should use the WAR file version and EAR file should use the EAR file 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 Server 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

29.4 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 file.
2. Add the JAR file to "`oaam-extensions\WEB-INF\lib`" folder.
3. Repackage `oracle.oaam.extensions.war`.
4. Deploy the JAR file.

For detailed instructions, see Chapter 7, "Using the OAAM Extensions Shared Library to Customize OAAM."

29.5 Device Fingerprinting

Stale Cookies

Question/Problem: How will OAAM behave in Flash cookie and secure cookie stale scenarios?

Answer/Solution: See *Oracle Fusion Middleware Administrator's Guide for Oracle Adaptive Access Manager*.

- **What if persistent cookies are disabled?**

  Oracle Adaptive Access Manager uses different pieces of information about a machine to develop the "device fingerprint". If persistent cookies are disabled, Oracle Adaptive Access Manager still has other information to use in identifying the machine.

  Each feature standing on its own is not sufficient to secure the session; it is the combination of device fingerprint, IP, location, time, behavioral analysis, behavioral analysis as it relates to past behavior, and others.

- **What if secure cookies are deleted?**

  Oracle Adaptive Access Manager's fingerprinting technology does not solely rely on one element. Oracle Adaptive Access Manager uses dozens of attributes to recognize and "fingerprint" the device you typically use to login, providing greater "coverage" for an institution's customer base. If secure cookies are missing or disabled, Oracle Adaptive Access Manager uses other elements such as Flash object and HTTP headers for device identification.

- **What if Flash is not enabled?**

  Oracle Adaptive Access Manager's fingerprinting technology does not solely rely on one element. Oracle Adaptive Access Manager uses dozens of attributes to recognize and "fingerprint" the device you typically use to login, providing greater "coverage" for an institution's customer base. If Flash is not enabled, Oracle Adaptive Access Manager uses other elements such as secure cookie and HTTP headers for device identification.

29.6 Device Registration

Device Registration

Question/Problem: The user has an option in the challenge questions registration page to register a device:

"Check to register the device that you are currently using as a safe device"

If he skipped during the registration flow, he does not seem to have an option later on from the user preferences page. Is there a way to turn it on?

Answer/Solution: Device registration is set up to ask the user to register the device during registration and when being challenged.

You can turn it on in the register questions page of user preferences by setting:

```
bharosa.uio.default.userpreferences.questions.registerdevice.enabled=true
```

Currently the central user preferences page only enables for unregistering devices.

The user can register the device during registration, but he is also given the option to register the device when being challenged.

Question/Problem: The registration of devices does not appear in the registration flow. Device ID policies have been imported into OAAM Admin.

Answer/Solution: Device registration is not enabled by default. To enable device registration, `bharosa.uio.default.registerdevice.enabled` should be set to `true`.

29.7 Failure Counter

For the auto failure counter increment to work, Client Type for updateAuthStatus must be set to 9 (Question/Answer).

29.8 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. 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.

29.9 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

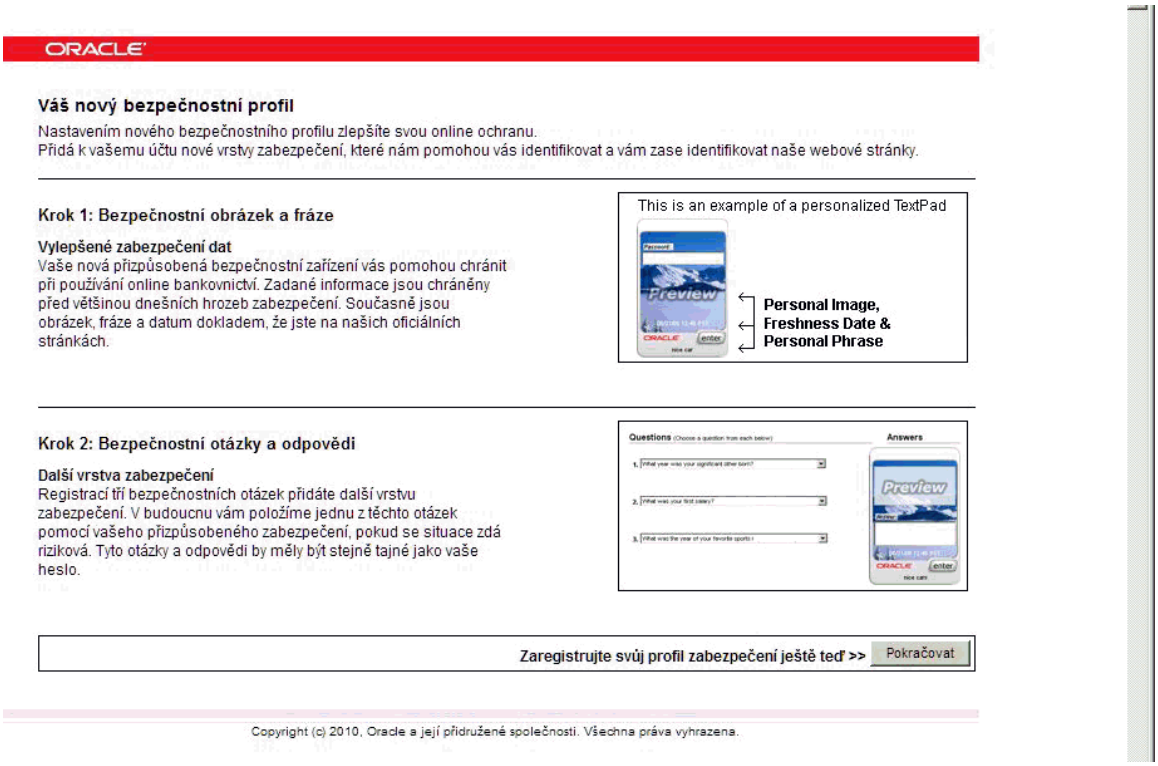
deploy them using product documentation. Any and all alterations to these images or the properties that correspond to them are considered custom development. Some issues to be careful of are text, hot spot, and key sizes. It is not recommended that these be made smaller than the provided samples.

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.

29.10 Man-in-the-Middle/Man-in-the-Browser

Question/Problem: I use mobile transaction authentication number to sign each transaction using an OTP through 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, verify 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.

29.11 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:

```
bharosa.uio.default.otp.generate.code.length = 5
bharosa.uio.default.otp.generate.code.characters = 1234567890
```

The pin generation method is in the base class (AbstractOTPChallengeProcessor), allowing integrators to override the generateCode method.

29.12 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.
- 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 will 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 website, 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 database 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 must 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 Management 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. The user has to use Apache 2.2.8 only.

Adding appid to HTTP Headers

Question/Problem: In `TestConfig.xml`, should I 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.

29.13 Virtual Authentication Devices

Tips and troubleshooting steps for OAAM 11g customization capability

These are the guidelines in Chapter 8, "Customizing OAAM Web Application Pages."

Question/Problem: I am trying to use the registerQuestionHTML instead of registerQuestion by putting the following entry in the property file. I had tried put it in both oaam_custom.properties and client_resource.properties file and I do not see it is taking the value. bharosa.uio.default.RegisterQuestions.authenticator.device = DeviceHTMLControl
bharosa.uio.default.ChallengeQuestion.authenticator.device=DeviceHTMLControl

Answer/Solution: The properties mentioned are only used if the AuthentiPad checkpoint is turned off, as mentioned in the property file. The property for authentipad checkpoint is bharosa.uio.default.use.authentipad.checkpoint=true. Alternatively (and recommended) would be to modify the policies in the authentipad checkpoint to have the desired device outcome for the page.

Tip: All user displayed strings should be customized in client_resource.properties or its locale specific variations (example: client_resource_es.properties)

Tip: Place custom user interface strings to WEB-INF/classes/client_resource.properties. Place custom frame file image to WEB-INF/classes/bharosa_properties/pad_images (because it has to be on classpath). Place frame file property to WEB-INF/classes/client_resource.properties and update the value for frame file, for example: "pad_images/authenticator_pad.png")

Tip: Custom JSPs cache issue: Oracle Weblogic seems to cache the custom JSP, so once the WAR file has been deployed with the JSP it is hard to see any additional changes to it. Some workarounds to get around this that are tested are by changing the file name (and property value) or by clearing the deployment directories used by Oracle Weblogic. ("DefaultDomain/servers/DefaultServer/tmp" and/or "o.j2ee/drs").

Disabling OAAM Authentication Pad

Question: Is there a way to disable OAAM authentipad when custom extensions war files are used?

For OAAM Server there are 2 options:

- Modify the authentipad policy to always return "OAAM HTML Pad".
- Set the following properties in OAAM Server:

```
bharosa.uio.default.use.authentipad.checkpoint=false
bharosa.uio.default.Password.authenticator.device=DeviceHTMLControl
bharosa.uio.default.Password.authenticator.device.upgraded=DeviceHTMLControl
bharosa.uio.default.ChallengeQuestion.authenticator.device=DeviceHTMLControl
bharosa.uio.default.RegisterQuestions.authenticator.device=DeviceHTMLControl
bharosa.uio.default.ChallengeSMS.authenticator.device=DeviceHTMLControl
```

```
bharosa.uio.default.ChallengeEmail.authenticator.device=DeviceHTMLControl
```

Another alternative is that as of 11.1.2 the login page can be consolidated to one page with the following properties:

> **Note:** To effect challenges, you need to perform one of the two options above.

```
bharosa.uio.default.login.auth.enabled=true
bharosa.uio.default.credentials.enum.password.enabled=true
bharosa.uio.default.signon.links.enum.wherepassword.enabled=false
```

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 through the screen reader and other assistive technologies.

You will need to modify `oaam_custom.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 `oaam_custom.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 specific 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 on-screen 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, the fraudster 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

Developing Custom Background Images

To develop custom background images for the virtual authentication devices the following must performed:

1. Process images to correct resolution for each pad being used.
2. Next you must add the images to correct directories for each virtual authentication device. TextPad images should be in the TextPad directory, and so on. The directory will be in the form `bharosa.image.dirlist={oracle.oaam.home}/oaam_images`. This will resolve to `"/scratch/user/Oracle/Middleware/Oracle_IDM1/oaam/oaam_images"`. In this directory there are three sub-directories named keypad, questionpad and textpad.

Disabling Date And Time Stamp Displayed In The Authentipad Image In .Net

1. To disable date and time stamp, comment out:

```
CreateAuthentiPad API
AuthPad.TimeStampText = DateTime.Now.ToString();
CreateQuestionPad API
TimeStampText = DateTime.Now.ToString();
```

2. To display Timestamp

   Example 1 (displays user defined string):

```
ret.AuthPad.TimeStampText = "monster";
ret.TimeStampText = "muppet";
```

   Example 2 (displays current time):

```
AuthPad.TimeStampText = DateTime.Now.ToString();
TimeStampText = DateTime.Now.ToString();
```

Changing the Limit of Characters for Passwords

To change the character limit for passwords entered in to OAAM server, update the value for the following property in the `oaam_cli.properties` file:

```
bharosa.authentipad.textpad.datafield.maxLength
```

For existing Access Manager and OAAM integration deployments, the value for the property can be updated using the OAAM Administration Console or shared library.

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 `oaam_custom.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 ensure that the backgrounds folder 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 folder.
- Check the properties file to ensure that the backgrounds folder setting is correct.
- Check that the system is configured to assign images for personalization.

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.

ASP.NET

ASP.NET is a Web application framework that allows programmers to build dynamic websites, Web applications, and Web services. OAAM provides an OAAM .NET development kit (SDK). The OAAM .NET SDK to use for integrating ASP.NET applications with OAAM. It includes the OAAM .NET APIs that are exposed by the OAAM .NET library, OAAM sample .NET applications, OAAM Flash movie page, and other files that are required for .NET native integration. ASP.NET applications, written in any ASP.NET language, can use the OAAM .NET API to call Oracle Adaptive Access Manager.

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 as follows:

- If a user logs in for the first time and 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, the user is challenged. The user 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?"

AuthUser

User in the authentication database.

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 this user 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, extensions 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 run-time 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 through 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, and others, used by a user

Device Fingerprinting

Device fingerprinting and identification is one of the many attributes OAAM utilizes to assess the risk of an access request or transaction. OAAM provides a full, layered security solution. Device fingerprinting and identification represents only one of the layers.

OAAM provides browser based fingerprinting in a pure web environment. This means that no client software is required, which makes deployment of the solution to large and diverse user populations manageable. As well, not placing any logic on the client ensures that there is no logic to be compromised. If however you want or need to deploy client software such as a java applet or native mobile application, OAAM can accept additional device fingerprinting data (IMEI, MAC, and so on) gathered by the client. Providing additional data such as hardware based unique identifiers increases the pertinence of the fingerprinting.

When using browser based fingerprinting each browser will have it's own device ID. This means the Internet Explorer and Firefox installed on the same laptop will appear as two distinct device IDs in the OAAM console. This is by design and has no impact on the level of security OAAM provides. This is because OAAM is profiling all the device IDs used in a deployment both in relation to users and independently. If it's typical for a user to sometimes use Firefox and sometimes use Internet Explorer OAAM tracks this as part of their profile.

OAAM browser fingerprinting logic utilizes the browser user agent string data along with other contextual data available in the session. This other data can include a one-time use secure cookie and Flash shared object if available. The device fingerprinting logic automatically deals with the situation where a user either deletes their cookies and FSO or does not have them enabled at all. OAAM will assign a new ID for a short period (3 logins) then revert back to first ID from there on if a user's behaviour is consistent (same user and IP for example). The device fingerprinting logic also accounts for common changes in device data such as an operating system or browser upgrade.

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 owning special knowledge

Entities

An entity structure is created by combining multiple related data points for optimization. The entity can be reused in multiple transactions by creating new

instances of the entity. Entities are not associated with or dependent on any transactions.

For example, `shipping address` and `billing address` instances can be created for two different transactions from the `Address` entity. The address entity can include `street number`, `street name`, `apartment number`, `city`, `state`, `postal code`, and `country` as its data points.

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, 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

In-Proc Integration

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 JAR files and configured properties, as appropriate.

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 is 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, and others, 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 user can use the account to log in to 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 through 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 preconfigured 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 through a configured out of band channel. Supported OTP delivery channels include short message service (SMS), eMail, and instant messaging. You can use OTP Anywhere to compliment KBA challenge or instead of KBA. As well you can use both OTP Anywhere and KBA alongside practically any other authentication type required in a deployment. Oracle Adaptive Access Manager also provides a challenge processor framework. You can use this

framework 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 personalized 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 website's traffic to another, bogus website.

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.

SAMPLE

An OAAM sample application is available for your reference. Before you integrate the APIs into your own application, be sure to download the OAAM sample application. It illustrates how to call the product APIs. It is available as a form of documentation. The OAAM sample application is not intended to be used as production code.

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 calculates the score produced by the different rules in a policy. The Policy Set Scoring Engine calculates 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.

Secure Cookie

The secure cookie stored by the OAAM in the client's browser is a tracking cookie:

- It does nott store any information about the user.

- It is only used to track if the user had logged in from this browser before to identify a device.
- It is valid for a single user only.

If OAAM is able to find this cookie in the browser, it compares this cookie with an expected value. If the two values match, it means that the request has come from a previously used device, hence the device ID is reused. If it does not match, it may be a stale or a modified cookie, so OAAM does not consider it. If the cookie is not present in the browser, it is a new request. In any case this cookie is discarded and a new cookie is generated.

Security Assertion Markup Language (SAML)

Security Assertion Markup Language (SAML) is an XML-based open standard for exchanging authentication and authorization data between security domains, that is, between an identity provider (a producer of assertions) and a service provider (a consumer of assertions).

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 with a customer trying to access their bank account). The token is used in addition to or for 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.

SOAP Service Wrapper API

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++.

This integration requires adding lightweight client libraries (JAR or DLL files) to the client library.

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 want. The personalized 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 maps the client/external transaction data to transactions in the OAAM 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, and others, 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