Oracle® FMW

Administering Oracle Advanced Authentication and Oracle Adaptive Risk Management

F52013-09
September 2023

Oracle FMW Administering Oracle Advanced Authentication and Oracle Adaptive Risk Management,

F52013-09

Copyright © 2021, 2023, Oracle and/or its affiliates.

Primary Author: Oracle Corporation

Contents

Preface

Administering Oracle Advanced Authentication (OAA) and Oracle Adaptive Risk Management (OARM) describes how to install Oracle Advanced Authentication and Oracle Adaptive Risk Management, configure and integrate with OAM and ORA to provide multi-factor authentication capabilities, and transition from Oracle Adaptive Access Manager (OAAM) to OARM and OAA.

Audience

This guide is intended for:

- Administrators responsible for installing and configuring Oracle Advanced Authentication (OAA)
- Administrators responsible for installing and configuring Oracle Adaptive Risk Management (OARM)
- Administrators responsible for integrating Oracle Access Management (OAM) and Oracle RADIUS Agent (ORA) with OAA for multi-factor authentication.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at https://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 https://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit https://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

For more information, refer to the following documents:

- Online Help
- Administering Oracle Access Management
- REST API documentation
- Administering Oracle RADIUS Agent

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.

List of Figures

Part I

Introduction to Oracle Advanced Authentication and Oracle Adaptive Risk Management

1

Introducing Oracle Advanced Authentication

Oracle Advanced Authentication (OAA) is a standalone microservice that can be used by applications to establish and assert the identity of users.

Topics

1.1 About Oracle Advanced Authentication (OAA)

Oracle Advanced Authentication (OAA) is a standalone micro-service that supports establishing and asserting the identity of users. It provides a comprehensive solution that is simple to deploy and use.

OAA provides strong authentication using Multiple Authentication Factors (MFA). A wide range of authentication (challenge) factors are available out-of-the-box for establishing the identity of users.

It supports integration with Oracle Access Management (OAM) and Oracle RADIUS Agent (ORA) to provide MFA capabilities.

1.2 Features of Oracle Advanced Authentication (OAA)

Oracle Advanced Authentication (OAA) constitutes unique features that facilitate deployment, configuration, and integration with other products.

The following are the features of OAA:

- Runs as a standalone micro-service on a Kubernetes platform and is deployed using Helm charts.
- Supports integration with the following clients to enable Multi-factor Authentication (MFA):
  - Clients providing web-based user login flows, such as Oracle Access Management (OAM). OAA integrates with OAM through Trusted Authentication Protocol (TAP).
  - Clients providing API-based user login flows, such as Oracle RADIUS Agent (ORA). OAA integrates with ORA through REST APIs. This type of integration enables clients to manage its own user-flow orchestration.
- Provides the `OAAAuthnPlugin` for integrating with OAM. The plug-in also enables migration of user data from the identity store on OAM to OAA.
- Provides web UI (OAA Administration console) for administrators to create and manage client registrations, assurance levels and rules. Administrators can also achieve all the administration tasks using REST APIs.

- Provides web UI (User Preferences console) for end-users to manage and register their challenge-factors. User self-registration and management can also be performed using REST APIs.
- Web UIs are secured by OAM OAuth and OpenID Connect (OIDC).
- Provides the following challenge-factors out-of-the-box:
  - TOTP (Time-based One Time Password) with Oracle Mobile Authenticator (OMA), Google, and Microsoft
  - OTP (One Time Password) with email and SMS
  - Yubikey OTP
  - FIDO2
  - Knowledge-Based Authentication (KBA)
  - Push Notifications

1.3 System Architecture and Components

OAA is composed of micro-services, web applications, platform abstractions, and authentication factor providers, along with an RDBMS used for storing user preferences and service data/metadata.

The components of OAA are as follows:

OAA Runtime and API

This component is the main processing unit of the system and provides REST APIs for managing user challenge flows and orchestrating the flow using challenge factors.

This runtime component integrates with API-based clients, for example, Oracle RADIUS Agent (ORA).

OAA Runtime UI

This component provides User Interface (UI) pages for managing the user challenge flow. For the end-user, it provides the user interface for choosing the challenge factor, and going back and forth with challenge factors during the flow.

This runtime component integrates with clients running browser-based flows, for example, Oracle Access Management (OAM), using OAuth and OpenID Connect (OIDC).

It provides the following UI Pages:

User Challenge Choice Pages: This renders the available challenges for users to choose from. It also provides an option to remember the choice the next time. After the user chooses the challenge, it redirects to the User Challenge Answer Page.

User Challenge Answer Pages for factors: The challenge answer page retrieves the answer from the chosen second factor specified by the user. Based on the type of challenge, the page provides a dialog box to type the answer in, for example, for the email, SMS, TOTP, and Knowledge-Based Authentication factors. If the challenge factor requires an assertion outside the browser, for example FIDO2, Yubikey, or push notifications, the page renders a timed wait. If verification fails it asks for the answer again, or sends the user back to choose another challenge, or times out. If verification succeeds users are redirected back to the agents. For more information about agents, see Understanding Oracle Advanced Authentication.

This page also allows the user to abandon the flow, or go back to the challenge choice page. It also gives users the option to remember the challenge choice for future requests, or allows that choice to be reset.

OAA Administration UI and API

This component provides REST APIs and Administration UI to manage integration agents, assurance levels, rules and groups. Rules are defined for each assurance level. Administrators can configure required challenge outcomes with the REST APIs or UI.

User Preferences UI and API

This component allows the end-user to see and manage their challenge factor registration using the UI or the user-preferences REST APIs.

Challenge Factors

Challenge factors are realized as services or containers that integrate with OAA runtime using REST API or the UI. Challenge factors can be configured using the UI or configuration API.

Persistent Store

This component is used for storing user preferences data and policy metadata. OAA supports database installation external to the Kubernetes cluster and provides the database schema to be imported.

Monitoring

Data monitoring is enabled for OAA service and policy management API.

1.4 Understanding Oracle Advanced Authentication

The following terms are used in OAA:

Integration Agent

In OAA, the clients that integrate with OAA are referred to as integration agents. The integration can be either REST-API-based, for example, Oracle RADIUS Agent (ORA) or browser-based through TAP, for example, Oracle Access Management (OAM).

Integration agents can be registered with OAA and managed through the Administration Console UI.

Assurance Level

Assurance Level indicates the level of assurance that is needed by the integration agent. It is a key contract between the integration agent and OAA that enforces the rules to be run for the user-login-flow. OAA runs the linked rules for that flow and determines Multi Factor Authentication (MFA) orchestration.

Assurance Levels can be defined to closely align with the NIST recommendations. However, this is not mandatory and Assurance Levels can be named in a reader-friendly way.

An integration agent can be assigned with multiple Assurance Levels, however, an Assurance Level can be associated with only one integration agent. Following are some examples of Assurance Levels:

- The RADIUS integration agent can define an Assurance Level named `Radius_DB12_AL` to indicate that the integration agent manages users from DB12 client
- OAM Server can define an Assurance Level named `OAM_AuthLevel6` to indicate that the resources are protected at auth-level 6 with OAM.
- OAM Server can define an Assurance Level named `PasswordLess1` to indicate that the resources are protected by a `Passwordless` scheme.

Challenge Factor

A Challenge Factor presents a challenge to the user and verifies if the user has correctly provided the expected input.

OAA supports the following factors out-of-the-box: E-Mail, SMS, Time-Based One Time Password (TOTP), FIDO2, Yubikey, Knowledge-Based Authentication (KBA), and Push notifications.

Rules

Each integration agent can have multiple assurance levels, and each assurance can have multiple rules in it. Each rule can have its own outcome of factors.

Rule: A Rule is an expression that contains attributes of user, such as UserID, IP address, and so on combined with conditions. At run time the actual values are substituted in this expression and the rule outcome as a group of actions is calculated.

Conditions: Conditions are expressions that compare the attributes with operators like equals, not equals, in group, and so on, based on the context.

2

Introducing Oracle Adaptive Risk Management

Oracle Adaptive Risk Management (OARM) is a comprehensive system that provides a way to monitor and control any user activity in your IT infrastructure (Single sign-on, Business Transactions).

Topics

2.1 About Oracle Adaptive Risk Management (OARM)

Oracle Adaptive Risk Management (OARM) is an integrated system that aggregates risk data associated with users and user activities, analyzes and evaluates business risks posed by users and their activities and provides advice to be acted on to mitigate them.

The system works best when integrated with Oracle Advanced Authentication (OAA), which executes risk mitigation actions to Block, Challenge, or Allow user activities based on the risk assessment associated with it.

The system can also work in a stand-alone mode where it can be consulted for remedial actions by consuming applications. OARM system is highly extensible owing to its micro-services based architecture, allowing additional capabilities to be added without having to indulge in a costly upgrade process.

2.2 Features of Oracle Adaptive Risk Management (OARM)

Oracle Adaptive Risk Management (OARM) system revolves around user activities, which are secured using business friendly rules.

OARM is shipped with an out-of-the-box User Authentication activity, which is baked in with a rich set of rules that can readily be used to secure the business. The system also provides the capability to augment the User Authentication activity with additional rules, remove rules not applicable to business, or add net new user activities to be monitored. OARM supports seeding data feeds from certified external sources that would also be used in risk analytics. This, combined with OARM's profiling capability provides the right mix of seed data for running analytics.

Configuration of Rules, managing and monitoring User Activities can be achieved with an intuitively designed Administration Console. The Administration Console allows Administrators to implement rules applicable to their Organization without being concerned with the nuances of the underlying system.

OARM in conjunction with OAA provides a large set of modern, multi-factor challenge methods allowing Administrators to choose challenge mechanisms that fit their business requirements. OAA also makes integration of OARM with existing Identity Management systems like Oracle Access Management Suite (OAM), very easy to achieve.

2.3 Understanding Terminologies in OARM

The following terminologies are used in OARM:

User Activity

User activity is any operation performed by the user that requires monitoring. For example, logging in, resetting passwords, and so on.

Out-of-the-box, OARM provides a user activity called **User Authentication**, which is built with a rich set of prepacked rules. **User Authentication** evaluates user activities to detect commonly found threats and take remedial actions or raise alerts.

Custom Activities

You can create your own custom activity in addition to the out-of-the-box **User Authentication** activity and create rules using the information collected from this custom activity. The rules are customized to the needs of the business. These rules could be transactional in nature, monitoring various aspects of the user activity in which the business is interested. Some examples of custom activities are internet banking or bill payment in a banking application. You can add rules that use information such as the amount of the payment, user information, and so on to identify a fraudulent money transfer.

Condition

Conditions are configurable evaluation statements that specifies one or more criteria to be satisfied by the access request in the OARM rule evaluation process and flow. They use datapoints from historical and runtime data to evaluate risk or business logic.

Each authentication rule contains one or more conditions that define whether a user is permitted or denied access to a protected resource by the rule. Conditions are pre-packaged in the system and cannot be created by a user. Conditions may take user inputs when adding them to a rule.

Rules

Rules are the main building blocks of decision making in OARM. Rules sum together the outcomes of various conditions that constitute them. Rules can then be used to make decisions to trigger actions or generate alerts.

You can implement new rules or edit existing rules based on new fraud data to fit business needs.

Action

An action is triggered based on the outcome of the evaluation of the configured rules. For example, actions can be forcing a user to register a security profile, blocking access, asking for a PIN or password based on a rule checking for Risky IPs. OARM provides several standard actions. The most prominent actions are `ALLOW`, `CHALLENGE`, and `BLOCK`.

Alert

Alerts are messages that indicate situations requiring attention based on the outcome of the evaluation of the configured rules. For example an alert is generated when a user logs in from a new country. Once the alert is issued, the Administrator can view the logged instance on the **User Sessions** page.

Groups

Groups are collections of similar items to simplify configuration workload. Groups can be used in places such as Rule conditions, Actions, and Alerts. You can choose from the following type of groups: User ID, User Name, Location, Device, Action, and Alert.

For example, to create a rule "Risky IP," you must add a condition to find out if the user IP used for login is in the list of risky IPs configured. The risky IPs are grouped together as Risky IP of type IP and the rule condition uses this group.

Profiles

Profiles 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 and used for calculating the current risk using rules.

Session

A session captures user's attributes and lifecycle, from the time of authentication until the resulting outcomes of the configured OARM risk management rules. An OARM session created is bound to both a user and the client with which they have authenticated.

OARM maintains a history of a user's sessions. Each session entry includes the Username, Device ID, IP address, and Session ID. You can view the session information using the **User Sessions** page.

The **User Sessions** page displays an overview of the events that transpired during a particular session for fraud analysis. It displays a summary of all the related information regarding the session, such as the session information, device information, location information from where the user logged in, user activities associated with the session, and rules, actions, and alerts triggered for the session.

OARM provides the capability to gather detailed information about the session and to allow you to drill down further into the details involved in the session. For example, you are a member of the security team at Acme Corp. You work with OARM on a regular basis, following up on escalated customer issues and security alerts. You perform a session search every couple hours throughout the day to identify any issues needing your attention.

2.4 OOTB User Authentication Rules Supported

OARM provides out-of-the-box (OOTB) authentication rules that alert you to potential attacker so that you can take corrective action.

The following table lists the OOTB user authentication rules supported by OARM.

Rule	Description
Block based on Risky IP	This rule will be triggered if an IP has previously been marked as a risky IP by the security team.
Block based on active anonymizer	This rule determines whether the IP address being used has been confirmed as an anonymizer within the last six months by the IP Location data provider.
Challenge based on Suspect Anonymizer	This rule determines whether the IP address being used has been confirmed as an anonymizer in the last two years but not in the last six months by the IP Location data provider.
Challenge based on Risky Device	This rule will be triggered if a device has previously marked as risky by the security team.
Challenge based on Country	This rule will be triggered if a user has logged in less than 20% of the time from this country in the last three months.
Challenge based on Less frequently used Autonomous System Number (ASN)	This rule will be triggered if a user activity occurs from a less frequently used Autonomous System Number (ASN).
Challenge based on Connection type	This rule will be triggered if a user has logged in with this connection type less than 6% of the time in the last month.
Challenge based on Routing type that is not utilized very often	This rule will be triggered if the user activity occurs via a less commonly used Routing type.
Challenge based on Least frequently used ISP	This rule will be triggered if user activity occurs from sparingly used ISPs.
Challenge based on Device	This rule will be triggered if a user has used this device to log in less than 10% of the time over the past month.
Challenge based on State from which least access happens	This rule will be triggered if user activity occurs from states with the least amount of activity.
Challenge based on Indicate Less Visited Time of day	This rule will be triggered if the user activity occurs at a rarely used time, such as 1 AM local time, when most users are dormant. This is a pattern-based authentication method in which an entity is a member of the pattern bucket less than a certain percentage of the time with all entities in the picture.
Challenge based on Browser locale from which least access happens	This rule is triggered if the user activity occurs in a browser locale with the least access. This is a pattern-based authentication method in which an entity is a member of the pattern bucket less than a certain percentage of the time with all entities in the picture.

Rule	Description
Challenge based on Connection type that is not utilized very often	This rule will be triggered if the user activity occurs via a less commonly used connection type. This is a pattern-based authentication method in which an entity is a member of the pattern bucket less than a certain percentage of the time with all entities in the picture.
Challenge based on Country from which least access happens	This rule will be triggered if the user activity occurs from states with the least amount of activity.
Challenge based on Day of week with the lowest number of visitors	This rule will be triggered if the user activity if the user activity occurs on the days of the week with the fewest visitors. This is a pattern-based authentication method in which an entity is a member of the pattern bucket less than a certain percentage of the time with all entities in the picture.
Challenge based on Risky countries	This rule will be triggered if a country has previously been marked as a risky country by the security team.
Challenge based on Unknown Anonymizer	There are currently no positive test results available. The initial anonymizer assignment is based on other sources and has yet to be confirmed by the IP Location data provider. This address is removed from the list if no positive test results are obtained.
Challenge based on Dormant Device	This rule will be triggered if a device has not been used in thirty days and more than two users login from it within twenty-four hours.
Challenge based on Device with many failures	This rule will be triggered if a device makes more than four unsuccessful login attempts within eight hours.
Challenge based on Maximum devices per user	This rule will be triggered if a user logs in using more than two devices within eight hours.
Challenge based on device maximum velocity	This rule will be triggered if a device appears to have traveled faster than jet speed in the last 20 hours since its last login.
Challenge based on risky connection type	This rule will be triggered if a connection type has previously been marked as a risky connection type by the security team.
Challenge based on limit activity from dormant IPs	This rule will be triggered if a dormant IP address is used excessively in a user activity.
Challenge based on based on limit user activity surge from an IP	This rule will be triggered if there is an increase in user activity from a specific IP address.

Rule	Description
Challenge based on based on private anonymizer	This IP address allegedly contains anonymous proxies that are not publicly accessible. As a result, automated tools cannot be used to test them on a regular basis. These addresses are typically associated with commercial ventures that provide anonymity services to the general public. Addresses with this designation are derived from ownership data or obtained from reliable sources.
Challenge based on user blocked recently	This rule will be triggered if a user has been blocked more than twice in the last eight hours.
Challenge based on maximum users per device	This rule will be triggered if more than four users log in using the same device within thirty days.
Challenge based on day of the week	This rule will be triggered if the user activity occurs on days of the week with the fewest visitors.
Challenge based on Time of day	This rule will be triggered if a user has accessed within the current time range less than 3% of the time in the last month.
Does user have a profile	This rule determines whether the pattern auto learning feature is enabled and whether the user has a historical behavior profile.
Is there enough pattern data available?	This rule determines whether there is enough pattern data available for auto-learning rules to use.
Predict if current session is fraudulent	This rule checks to see if the current session is predicted to be fraudulent using the Oracle Data Miner fraud classification model.
Predict if current session is anomalous	This rule predicts whether the current session is anomalous based on the anomaly ODM model.

2.5 Understanding the Sequence of User Activity Runtime API Calls

This section demonstrates how OARM processes user activities and provides values for API operations from the client application.

The following is a general sequence of API calls.

1. Create an OARM session using createSession (session-POST API). It creates a `requestId`, which is required for **Create Custom User Activity** API.
2. Client application then provides information about Custom User Activity by invoking Create Custom User Activity API (which uses the `request ID` created in Step 1).
3. Review to make sure the status of the **Create Custom User Activity** is successful before obtaining the transaction ID from the response.
4. Client can then 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.

5. Based on the outcome of the **processRules** API call, the client application can choose to call the Update Custom User Activity API to set the transaction status or to update data in the existing transaction.

   Note:

   Ensure that the **Custom User Activity** status is updated. This is due to the fact that some rules may use the status of previous transaction (user activity) as a data point.

6. In some cases, client applications can choose to execute a **processRules** API with a Pre Transaction checkpoint first and then Post Transaction kind of checkpoint that has policies/rules that have to be executed after a transaction is created. This can help application to figure out if transaction is good to execute, and then after execution any additional rules that may be required.

Part II

Installing Oracle Advanced Authentication and Oracle Adaptive Risk Management

Oracle Advanced Authentication (OAA) and Oracle Adaptive Risk Management (OARM) can be installed as standalone products or can be installed together. The following modes of installation are supported:

- OAA-OARM installation
- OAA only installation
- OARM only installation

The procedure for installing any of the three modes is the same. The following section provides details about installing OAA and OARM:

Procedure for Installing OAA and OARM

Note:

The installation instructions in this chapter explain how to configure OAA and OARM on a Kubernetes cluster where no other Oracle Identity Management products will be deployed. If you are installing OAA and OARM with other Oracle Identity Management products on the same Kubernetes cluster, you must follow the installation instructions in the Enterprise Deployment Guide. See Enterprise Deployment Guide for Oracle Identity and Access Management in a Kubernetes Cluster.

3

Procedure for Installing OAA and OARM

Topics

3.1 About the Management Container

The **Management Container** is a container that includes all the required scripts and tools needed to install OAA and OARM on a new or existing Kubernetes cluster.

This container runs as a pod in the Kubernetes cluster. It is not part of the OAA and OARM deployment itself, but facilitates deploying OAA and OARM to the Kubernetes cluster.

The Management Container pod has the following binaries installed based on `oraclelinux`, along with the standard linux utilities such as zip, iputils, net-tools, and vim:

- kubectl
- helm
- sqlplus: instantclient_19_10
- openssl

For more information about the Management Container, see the following topics:

- Components of the Management Container
- Preset Environment Variables in Management Container
- Mounted Volumes in the Management Container

3.1.1 Components of the Management Container

This section provides an overview of important files and folders in the Management Container pod.

Table 3-1 Management Container Files and Folder Reference

Files and Folders	Description
`OAA.sh`	This script file is used to install OAA and OARM. The `installOAA.properties` file must be given as an argument to the script for installing OAA, OAA-OARM, and OARM. For more information, see Preparing the Properties file for OAA and OARM Installation
`installsettings`	This folder contains the `oaaoverride.yaml` that can be customized to set the `replicaCount` for some of the services in OAA and OARM. To enable this you must set the `common.deployment.overridefile` property in the `installOAA.properties`.
`helmcharts`	This folder contains helm charts and values.yaml for all OAA and OARM services.
`libs`	This folder contains the following files: • `OAAAuthnPlugin.jar`: this plugin is used for integrating OAM with OAA. – If your OAM is pre April 22 Bundle patch (12.2.1.4.220404). The plugin must be imported into OAM. – If your OAM has April 22 Bundle patch (12.2.1.4.220404) or later then the plugin is included in OAM by default. For details, see Install the OAA Plugin for OAM • `messagingprovider-interface-install-oaa-<release-version>.jar`: This file can be used to customize the SMS and email factors in OAA. For more information, see Customizing Email and SMS Messaging Provider
`logs`	This folder maps to the NFS volume `<NFS_LOG_PATH>` and stores logs and status of the OAA and OARM installation.
`oaa_cli`	This folder contains files that can be customized and used to install geo-location data for OARM. For more information, see Loading Geo-Location Data
`scripts/creds`	This folder maps to the NFS volume `<NFS_CREDS_PATH>` and contains the following files used during installation: • trust.p12 • cert.p12 • k8sconfig • helmconfig

Table 3-1 (Cont.) Management Container Files and Folder Reference

Files and Folders	Description
`scripts/settings`	This folder maps to the NFS volume `<NFS_CONFIG_PATH>` and stores `installOAA.properties`, and `oaaoverride.yaml` configuration files required for installation.
`service/store/oaa/`	This folder maps to the NFS volume `<NFS_VAULT_PATH>` that is shared between management container and the OAA and OARM deployment. It stores the file based vault (if not using OCI based vault).

3.1.2 Preset Environment Variables in Management Container

The Management Container pod is configured with a predefined set of environment variables.

Table 3-2 Preset Environment Variables

Environment Variable	Description
`HELM_CONFIG`	This is set to `/u01/oracle/scripts/creds/helmconfig`.
`KUBECONFIG`	This is set to `/u01/oracle/scripts/creds/k8sconfig`.
`SCRIPT_PATH`	This is set to `/u01/oracle/scripts`. This contains the installation scripts.
`CONFIG_DIR`	This is a NFS volume `<NFS_CONFIG_PATH>` used to store the configuration externally. It is mounted to the path `/u01/oracle/scripts/settings` in the container.
`CREDS_DIR`	This is a NFS volume `<NFS_CREDS_PATH>` used to store credentials, such as helm config, kube config, and login private keys. It is mounted to the path `/u01/oracle/scripts/creds` in the container.
`LOGS_DIR`	This is a NFS volume `<NFS_LOGS_PATH>` used to store installation logs and status. It is mounted to path `/u01/oracle/logs` in the container.
`HELM_CHARTS_PATH`	This is the path where all the helm charts related to the installation exist.
`LD_LIBRARY_PATH`	Sets the instantclient folder. The variable is required to run the `sqlplus` and DB-related commands from instantclient present in the container.

Table 3-2 (Cont.) Preset Environment Variables

Environment Variable	Description
`LIBS_DIR`	This exists in the path `/u01/oracle/libs`. It contains the jar file required for customizing email and SMS providers and the OAM Authentication plugin. It also contains jars that are required for file based vault deployment.
`JARPATH`	This contains the jars required for file based vault to run properly.

3.1.3 Mounted Volumes in the Management Container

This section provides details about the mounted volumes in the Management Container pod.

Table 3-3 Mounted Volumes in Management Container

Mount Folder	Description	Permissions to be Set
`/u01/oracle/logs`	**Path not configurable.** This is used to store installation logs and status. This maps to NFS volume `<NFS_LOG_PATH>` created as a prerequisite.	**Read-Write-Execute** The NFS volume `<NFS_LOG_PATH>` must have Read-Write-Execute permissions for all.
`/u01/oracle/scripts/settings`	**Path not configurable.** This is used to store the customized configuration file for installing OAA and OARM. This maps to NFS volume `<NFS_CONFIG_PATH>` created as a prerequisite.	**Read-Write-Execute** The NFS volume `<NFS_CONFIG_PATH>` must have Read-Write-Execute permissions for all.
`/u01/oracle/scripts/creds`	**Path not configurable.** This is used to store credential files such as k8sconfig, helmconfig, trust.p12 and cert.p2. This maps to NFS volume `<NFS_CREDS_PATH>` created as a prerequisite.	**Read-Write-Execute** The NFS volume `<NFS_CREDS_PATH>` must have Read-Write-Execute permissions for all.
`/u01/oracle/service/store/oaa`	**Path is configurable.** This is used to store the vault artifacts for file-based vault. This maps to NFS volume `<NFS_VAULT_PATH>` created as a prerequisite.	**Read-Write-Execute** The NFS volume `<NFS_VAULT_PATH>` must have Read-Write-Execute permissions for all.

For more details about NFS volume requirements, see Configuring NFS Volumes.

3.2 Prerequisite Configurations for Installing OAA and OARM

Before progressing to the installation steps, ensure you have performed the following:

- Configuring a Kubernetes Cluster
- Configuring NFS Volumes
- Installing an Oracle Database
- Installing and Configuring OAM OAuth
- Setting Up Users and Groups in the OAM Identity Store
- Installing a Container Image Registry (CIR)
- Configuring CoreDNS for External Hostname Resolution
- Installation Host Requirements
- Generating Server Certificates and Trusted Certificates
- Create a Kubernetes Namespace and Secret

3.2.1 Configuring a Kubernetes Cluster

OAA and OARM are designed to be deployed on a Cloud Native Environment. OAA and OARM is composed of multiple components that run as microservices on a Kubernetes cluster, managed by Helm charts. Specifically, each component (microservice) is composed as a Kubernetes Pod, which is deployed to a Kubernetes Node in the cluster.

You must install a Kubernetes cluster that meets the following requirements:

- The Kubernetes cluster must have a minimum of three nodes.
- The nodes must meet the following system minimum specification requirements:

System	Minimum Requirements
Memory	64 GB RAM
Disk	150 GB
CPU	8 x CPU with (Virtualization support. For example, Intel VT)

- An installation of Helm is required on the Kubernetes cluster. Helm is used to create and deploy the necessary resources.
- A supported container engine must be installed and running on the Kubernetes cluster.
- The Kubernetes cluster and container engine must meet the minimum version requirements outlined in Document ID 2723908.1 on My Oracle Support.
- The nodes in the Kubernetes cluster must have access to a shared volume such as a Network File System (NFS) mount. Ths NFS mounts are used by the Management Container pod during installation, during runtime for the File Based Vault (if not using OCI based vault), and for other post installation tasks such as loading geo-location data.

3.2.2 Configuring NFS Volumes

All nodes in the Kubernetes cluster require access to a shared volumes on an NFS server. During the OAA/OARM installation, the Management container pod stores configuration information, credentials and logs in the NFS volumes. Once the installation is complete the pods require access to a volume that contains the File based vault (if not using OCI based vault) for storing and accessing runtime credentials.

The following NFS volumes must be created prior to the installation. In all cases the NFS export path must have read/write/execute permission for all. Make sure the NFS volumes are accessible to all nodes in the cluster.

Volume	Description	Path
Configuration	A NFS volume which stores the OAA configuration such as `installOAA.properties`.	<NFS_CONFIG_PATH>
Credentials	A NFS volume which stores OAA credentials such as Kubernetes and Helm configuration, SSH key, or PKCS12 files.	<NFS_CREDS_PATH>
Logs	A NFS volume which stores OAA installation logs and status.	<NFS_LOGS_PATH>
File based vault	A NFS volume which stores OAA runtime credentials.	<NFS_VAULT_PATH>

Note:

The NFS Server IP address and PATH's will be set in the `installOAA.properties`. See Preparing the Properties file for OAA and OARM Installation

3.2.3 Installing an Oracle Database

OAA and OARM uses a database schema to store information. You must install and configure an Oracle Database either on OCI or on-premises. The database must support partitioning feature/capabilities.

OAA and OARM supports Oracle Database 12c (12.2.0.1+), 18c, and 19c. For more detailed information on supported database versions, see http://www.oracle.com/technetwork/middleware/ias/downloads/fusion-certification-100350.html.

The Kubernetes cluster where OAA/OARM is to be installed, must have network connectivity to the database.

Note:

If using a non ASM database, you must make sure that the database has the parameter `DB_CREATE_FILE_DEST` set. For example:

```
SQL> connect SYS/<password> as SYSDBA;
Connected.
SQL> show parameter DB_CREATE_FILE_DEST;

NAME                                 TYPE        VALUE
------------------------------------ -----------
------------------------------
db_create_file_dest                  string      /u01/app/oracle/
oradata
```

If the parameter is not set, run the following:

```
SQL> ALTER SYSTEM SET DB_CREATE_FILE_DEST = '/u01/app/oracle/
oradata' scope=both;
```

where `/u01/app/oracle/oradata` is the path where your datafiles reside.

3.2.4 Installing and Configuring OAM OAuth

OAA and OARM need access to an Oracle Access Management (OAM) installation with OAuth enabled. The Kubernetes cluster where OAA/OARM is to be installed, must have network connectivity to the OAM installation.

The User Interface (UI) components of OAA and OARM (the OAA Administration Console and User Preferences console) are protected by Oracle Access Management (OAM) OAuth. The required OAuth components (Identity Domain, Resource, Client) can be configured during the OAA installation. However, the following are the prerequisites steps to be performed before the installation can configure the required OAuth components in OAM:

Note:

You can skip the OAuth configuration in this section if the UI components are not required or need to be disabled during the installation. If skipping OAuth configuration you must set `oauth.enabled=false` along with associated properties in `installOAA.properties`. For more details, see OAM OAuth Configuration.

1. Install Oracle Access Management. For details, see Installing Oracle Access Management.
2. Register WebGates with OAM. For details, see Registering and Managing OAM Agents.
3. Enable OAuth on the Oracle Access Management Console:
   a. Log in to the OAM Console `https://<OAMAdminHost>:<OAMAdminPort>/oamconsole/`
   b. From the Welcome page, click **Configuration** and then click **Available Services**

   c. Click **Enable Service** beside OAuth and OpenIDConnect Service (or confirm that the green status check mark displays).

4. Open the **mod_wl_ohs.conf** file located at `<OHS_HOME>/user_projects/domains/base_domain/config/fmwconfig/components/OHS/<ohs_instance_name>` and add the following:

```
<Location /oauth2>
SetHandler weblogic-handler
WebLogicHost <OAM_Managed_Server_Host>
WebLogicPort <OAM_Managed_Server_Port>
</Location>

<Location /oam>
 SetHandler weblogic-handler
 WebLogicHost <OAM_Managed_Server_Host>
 WebLogicPort <OAM_Managed_Server_Port>
</Location>

<Location /.well-known/openid-configuration>
 SetHandler weblogic-handler
 WebLogicHost <OAM_Managed_Server_Host>
 WebLogicPort <OAM_Managed_Server_Port>
 PathTrim /.well-known
 PathPrepend /oauth2/rest
</Location>

<Location /.well-known/oidc-configuration>
SetHandler weblogic-handler
WebLogicHost <OAM_Managed_Server_Host>
WebLogicPort <OAM_Managed_Server_Port>
PathTrim /.well-known
PathPrepend /oauth2/rest
</Location>

<Location /CustomConsent>
SetHandler weblogic-handler
WebLogicHost <OAM_Managed_Server_Host>
WebLogicPort <OAM_Managed_Server_Port>
</Location>
```

Note:

`<OAM_Managed_Server_Host>` and `<OAM_Managed_Server_Port>` is the host name and port of the OAM managed server.

5. Open the **httpd.conf** file located at `<OHS_HOME>/user_projects/domains/base_domain/config/fmwconfig/components/OHS/<ohs_instance_name>/` and add the following:

Note:

Specify a value for your OAuth Identity Domain in `<DomainName>`. The `<DomainName>` will be used later in the parameter `oauth.domainname` in the `installOAA.properties`.

```
<IfModule mod_rewrite.c>
RewriteEngine on
RewriteRule ^/oauth2/rest/authorize? /oauth2/rest/authorize?
domain=<DomainName> [PT,QSA,L]
RewriteRule ^/oauth2/rest/token? /oauth2/rest/token?domain=<DomainName>
[PT,QSA,L]
RewriteRule ^/oauth2/rest/token/info? /oauth2/rest/token/info?
domain=<DomainName> [PT,QSA,L]
RewriteRule ^/oauth2/rest/authz? /oauth2/rest/authz?domain=<DomainName>
[PT,QSA,L]
RewriteRule ^/oauth2/rest/userinfo? /oauth2/rest/userinfo?
domain=<DomainName> [PT,QSA,L]
RewriteRule ^/oauth2/rest/security? /oauth2/rest/security?
domain=<DomainName> [PT,QSA,L]
RewriteRule ^/oauth2/rest/userlogout? /oauth2/rest/userlogout?
domain=<DomainName> [PT,QSA,L]
</IfModule>

<IfModule mod_headers.c>
#Add Identity domain header always for OpenID requests
RequestHeader set X-OAUTH-IDENTITY-DOMAIN-NAME "<DomainName>"
</IfModule>
```

6. For the OHS WebGate defined in the previous steps, perform the following in the OAM console:
   a. Create each of the following resources and set the **Protection Level** as `Excluded`.
      - /oauth2/rest/**
      - /oam/**
      - /.well-known/openid-configuration
      - /iam/access/binding/api/v10/oap/**
      - /oam/services/rest/**
      - /iam/admin/config/api/v1/config/**
      - /oaa-admin/**
      - /admin-ui/**
      - /oaa/**
      - /policy/**
      - /oaa-policy/**
      - /oaa-email-factor/**
      - /oaa-sms-factor/**

- /oaa-totp-factor/**
- /oaa-yotp-factor/**
- /fido/**
- /oaa-kba/**
- /oaa-push-factor/**
- /risk-analyzer/**
- /risk-cc/**
- /consolehelp/**
- /otpfp/**

b. Create each of the following resources and set the **Protection Level** as `Protected` and set the **Authentication Policy** and **Authorization Policy** as `Protected Resource Policy`

- /oauth2/rest/approval (this is for `POST` operation)
- /oam/pages/consent.jsp (this is for `GET` operation)

For more information, see Adding and Managing Policy Resource Definitions

7. Configure the OHS as reverse proxy in OAM. To do this:

a. Log in to the OAM Console `https://<OAMAdminHost>:<OAMAdminPort>/oamconsole/`

b. From the Welcome page, click **Configuration** and in the **Settings** tile, click **View** > **Access Manager**.

c. Under **Load Balancing** specify the **OHS Host** and **OHS Port**.

3.2.5 Setting Up Users and Groups in the OAM Identity Store

Oracle Advanced Authentication requires the following groups to be configured in LDAP:

- **OAA-Admin-Role**, which is used to authenticate users who have permission to access the OAA Administration Console UI.
- **OAA-App-User**, which contains users who have permission to access the OAA User Preferences UI.

Every user who needs OAA access must be a member of the `OAA-App-User` group, otherwise they will not be able to log in to the OAA User Preferences UI through OAM OAuth. Similarly, for the administrator to be able to access the OAA Administration console, they must be a member of the `OAA-Admin-Role` group.

Note:

A user cannot be a member of both the OAA-Admin-Role and OAA-App-User groups. Therefore, it is recommended that you have a dedicated administrator user name.

Creating Users and Groups

To create the users and groups:

1. Create an LDIF file `oaa_admin.ldif` with the following contents:

   > **Note:**
   >
   > The following example is for an OAM enabled directory.

   ```
   dn: cn=oaaadmin,cn=Users,dc=example,dc=com
   changetype: add
   objectClass: orclUserV2
   objectClass: oblixorgperson
   objectClass: person
   objectClass: inetOrgPerson
   objectClass: organizationalPerson
   objectClass: oblixPersonPwdPolicy
   objectClass: orclAppIDUser
   objectClass: orclUser
   objectClass: orclIDXPerson
   objectClass: top
   objectClass: OIMPersonPwdPolicy
   givenName: oaaadmin
   uid: oaaadmin
   orclIsEnabled: ENABLED
   sn: oaaadmin
   userPassword: <Password>
   mail: oamadmin@example.com
   orclSAMAccountName: oaaadmin
   cn: oaaadmin
   obpasswordchangeflag: false
   ds-pwp-password-policy-dn:
   cn=FAPolicy,cn=pwdPolicies,cn=Common,cn=Products,cn=OracleContext,dc=examp
   le,dc=com

   dn:cn=OAA-Admin-Role,cn=Groups,dc=example,dc=com
   changetype: add
   objectClass: top
   objectClass: groupofuniquenames
   uniqueMember: cn=oaaadmin,cn=Users,dc=example,dc=com

   dn:cn=OAA-App-User,cn=Groups,dc=example,dc=com
   changetype: add
   objectClass: top
   objectClass: groupofuniquenames
   ```

2. Load the LDIF file into the directory. The following example assumes you are using Oracle Unified Directory:

   ```
   $ cd INSTANCE_DIR/OUD/bin
   ldapmodify -h <OUD_HOSTNAME> -p 1389 -D "cn=Directory Manager" -w
   <password> -f oaa_admin.ldif
   ```

Adding Existing Users to the OAA User Group

If you want to configure existing users to use OAA, you need to add them to the `OAA-App-User` group created above:

1. Run the following commands in the LDAP instance. These commands create an LDIF file that adds all your existing users to the `OAA-App-User` group:

   ```
   echo "dn:cn=OAA-App-User,cn=Groups,dc=example,dc=com" >
   update_group.ldif


   echo "changetype: modify" >> update_group.ldif


   echo "add: uniqueMember" >> update_group.ldif


   ldapsearch -h <OUD_HOSTNAME> -p 1389 "cn=Directory Manager" -w
   <password> -b cn=Users,dc=example,dc=com "cn=*" dn | grep -v
   oaaadmin | grep -v "dn: cn=Users,dc=example,dc=com" | grep cn| awk
   ' { print "uniqueMember: "$2 } ' >> update_group.ldif
   ```

2. Edit the `update_group.ldif` and remove any users you don't want to add to the group.
3. Load the LDIF file into the directory:

   ```
   ldapmodify -h <OUD_HOSTNAME> -p 1389 -D "cn=Directory Manager" -w
   <password> -f update_group.ldif
   ```

3.2.6 Installing a Container Image Registry (CIR)

During the Management Container installation, OAA and OARM container images are pushed to a Container Image Registry (CIR). When OAA and OARM is deployed the deployment pulls the images from the same Container Image Registry. You must therefore install a Container Image Registry as a prerequisite. The Container Image Registry must be accessible from all nodes in the Kubernetes cluster where OAA/OARM is to be deployed.

Depending on the CIR you are using, you may have to create the following repository entries in the CIR prior to installation of OAA and OARM. For example, if using Oracle Container Registry in Oracle Cloud Infrastructure (OCI) you must create these repository entries in advance, otherwise the install will fail to push the images :

- oaa-admin
- oaa-factor-email

- oaa-factor-fido
- oaa-factor-kba
- oaa-factor-push
- oaa-factor-sms
- oaa-factor-totp
- oaa-factor-yotp
- oaa-factor-custom
- oaa-mgmt
- oaa-policy
- oaa-spui
- oaa-svc
- risk-cc
- risk-engine

If you do not have a CIR, you can download Docker Registry from: https://hub.docker.com/_/registry/.

3.2.7 Configuring CoreDNS for External Hostname Resolution

In order for the Kubernetes cluster to resolve the required hostnames for the installation, you must configure CoreDNS in your cluster.

You must configure CoreDNS as follows:

- Either add the hostname.domain and IP addresses of any Proxy Severs, the Kubernetes nodes, the OAM OAuth server, the Oracle Database, and your Container Image Registry; or
- Add the Domain Names Servers (DNS) that can resolve the hostname.domain and IP addresses of any Proxy Severs, the Kubernetes nodes, the OAM OAuth server, the Oracle Database, and your Container Image Registry.

Note: The instructions below are generic for Kubernetes and may not be applicable to all Kubernetes vendors. Refer to your Kubernetes vendor specific documentation on how to configure CoreDNS.

Adding individual hostnames and IP addresses or DNS to CoreDNS

1. Run the following command to edit the coredns configmap:

   ```
   kubectl edit configmap/coredns -n kube-system
   ```

   This will take you into an edit session similar to `vi`.
2. If you prefer to add each individual hostname and IP address, add a hosts section to the file including one entry for each of the hosts you wish to define. For example:

   ```
   apiVersion: v1
   data:
     Corefile: |
       .:53 {
   ```

```
        errors
        health {
           lameduck 5s
        }
        ready
        kubernetes cluster.local in-addr.arpa ip6.arpa {
           pods insecure
           fallthrough in-addr.arpa ip6.arpa
           ttl 30
        }
        prometheus :9153
        forward . /etc/resolv.conf {
           max_concurrent 1000
        }
        cache 30
        loop
        reload
        loadbalance
        hosts custom.hosts example.com {
             1.1.1.1 oam.example.com
             1.1.1.2 db.example.com
             1.1.1.3 container-registry.example.com
             1.1.1.4 masternode.example.com
             1.1.1.5 worker1.example.com
             1.1.1.6 worker2.example.com
             fallthrough
        }
      }
kind: ConfigMap
metadata:
  creationTimestamp: "2021-11-09T14:08:31Z"
  name: coredns
  namespace: kube-system
  resourceVersion: "25242052"
  uid: 21e623cf-e393-425a-81dc-68b1b06542b4
```

Alternatively, if you prefer to add the Domain Name Server (DNS) then add a section for the DNS:

```
apiVersion: v1
data:
  Corefile: |
    .:53 {
        errors
        health {
           lameduck 5s
        }
        ready
        kubernetes cluster.local in-addr.arpa ip6.arpa {
           pods insecure
           fallthrough in-addr.arpa ip6.arpa
           ttl 30
        }
        prometheus :9153
```

```
            forward . /etc/resolv.conf {
               max_concurrent 1000
            }
            cache 30
            loop
            reload
            loadbalance
        }
        example.com:53 {
          errors
          cache 30
          forward . <DNS_IPADDRESS>
          }
kind: ConfigMap
metadata:
  creationTimestamp: "2021-11-09T14:08:31Z"
  name: coredns
  namespace: kube-system
  resourceVersion: "25242052"
  uid: 21e623cf-e393-425a-81dc-68b1b06542b4
```

3. Save the file (`!wq`).
4. Restart CoreDNS:
   a. Run the following command to restart coredns:

      ```
      kubectl delete pod --namespace kube-system --selector k8s-app=kube-dns
      ```

   b. Ensure the coredns pods restart without any problems by running the following command:

      ```
      kubectl get pods -n kube-system
      ```

      If any errors are shown use the following command to view the logs, then correct by editing the coredns configmap again:

      ```
      kubectl logs -n kube-system coredns--<ID>
      ```

Validating DNS Resolution

Most containers do not have built in networking tools to allow you to check that the configuration changes you made are correct. The easiest way to validate the changes is to use a lightweight container with the network tools installed, such as **alpine**.

1. Run the following command to run an alpine container:

   ```
   kubectl run -i --tty --rm debug --image=docker.io/library/alpine:latest --
   restart=Never -- sh
   ```

   This will take you inside a bash shell in the container.

2. Inside the container you can then run `nslookup` against the Database, OAM OAuth Server, Container Image Registry etc, for example:

   ```
   nslookup oam.example.com
   ```

3.2.8 Installation Host Requirements

The Management Container installation can take place from any node that has access to deploy to the Kubernetes cluster. This section lists the specific requirements for the node where the installation of the Management Container will take place.

The installation host must meet the following requirements:

- Linux x86_64.
- A minimum of 2 x CPU's and 16GB RAM.
- At least 40GB of free space in the root partition "`/`".
- The node must have access to deploy to the Kubernetes cluster where the Management Container and OAA/OARM will be installed. The kubectl version requirements are the same as per Configuring a Kubernetes Cluster.
- Podman 3.3.0 or later. (If podman is not an option, Docker 19.03 or later can be used).
- Helm 3.5 or later.
- Openssl.
- If your environment requires proxies to access the internet, you must set the relevant proxies in order to connect to the Oracle Container Registry. For example:

  ```
  export http_proxy=http://proxy.example.com:80
  export https_proxy=http://proxy.example.com:80
  export HTTPS_PROXY=http://proxy.example.com:80
  export HTTP_PROXY=http://proxy.example.com:80
  ```

  You must also make sure that `no_proxy` is set and includes the nodes referenced in the output under `server` in `kubectl config view`. For example if `kubectl config view` shows:

  ```
  kubectl config view
  apiVersion: v1
  clusters:
  - cluster:
      certificate-authority-data: DATA+OMITTED
      server: https://masternode.example.com:6443
    name: kubernetes
  contexts:
  etc...
  ```

  then set the following:

  ```
  export NO_PROXY=masternode.example.com:$NO_PROXY
  export no_proxy=masternode.example.com:$no_proxy
  ```

- The node must have access to your Container Image Registry as per Installing a Container Image Registry (CIR).
- In order for the installation to pull supporting images, the Administrator performing the install must have login credentials for Oracle Container Registry. You will be prompted for these credentials during the installation of the Management Container.
- Make sure you can login to Oracle Container Registry from the installation host:

  ```
  podman login container-registry.oracle.com
  ```

  Note:

  If you are are not using podman and are using Docker then run: `docker login container-registry.oracle.com`.

3.2.9 Generating Server Certificates and Trusted Certificates

OAA and OARM uses SSL for communication. You must generate server certificate and trusted certificate keystores (PKCS12) prior to installation.

Follow the relevant section below depending on whether you are using a third party Certificate Authority for generating your certificates, or if you want to generate your own CA and certificates for testing purposes.

Using a third party CA for generating certificates

1. On the node where the Management container installation will be run from, create a directory and navigate to that folder, for example:

   ```
   mkdir <workdir>/oaa_ssl
   export WORKDIR=<workdir>
   cd $WORKDIR/oaa_ssl
   ```

2. Generate a 4096 bit private key (`oaa.key`) for the server certificate:

   ```
   openssl genrsa -out oaa.key 4096
   ```

3. Create a Certificate Signing Request (`oaa.csr`):

   ```
   openssl req -new -key oaa.key -out oaa.csr
   ```

   When prompted enter details to create your Certificate Signing Request (CSR). For example:

   ```
   You are about to be asked to enter information that will be incorporated
   into your certificate request.
   What you are about to enter is what is called a Distinguished Name or a
   DN.
   There are quite a few fields but you can leave some blank
   For some fields there will be a default value,
   If you enter '.', the field will be left blank.
   ```

```
-----
Country Name (2 letter code) [XX]:US
State or Province Name (full name) []:California
Locality Name (eg, city) [Default City]:Redwood City
Organization Name (eg, company) [Default Company Ltd]:Example
Company
Organizational Unit Name (eg, section) []:Security
Common Name (eg, your name or your server's hostname)
[]:oaa.example.com
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

4. Send the CSR (`oaa.csr`) to the third party CA.
5. Once you receive the certificate from the CA, rename the file to `oaa.pem` and copy it to the `$WORKDIR/oaa_ssl` directory.

   Note:

   The certificate `oaa.pem` needs to be in PEM format. If not in PEM format convert it to PEM using openssl. For example, to convert from DER format to PEM:

   ```
   openssl x509 -inform der -in oaa.der -out oaa.pem
   ```

6. Copy the Trusted Root CA certificate (`rootca.pem`), and any other CA certificates in the chain (`rootca1.pem`, `rootca2.pem`, etc) that signed the `oaa.pem` to the `$WORKDIR/oaa_ssl` directory. As per above, the CA certificates must be in PEM format, so convert if necessary.
7. If your CA has multiple certificates in a chain, create a `bundle.pem` that contains all the CA certificates:

   ```
   cat rootca.pem rootca1.pem rootca2.pem >>bundle.pem
   ```

8. Create a Trusted Certificate PKCS12 file (`trust.p12`) from the files as follows:

   ```
   openssl pkcs12 -export -out trust.p12 -nokeys -in bundle.pem
   ```

   When prompted enter and verify the Export Password.

   Note:

   If your CA does not have a certificate chain replace `bundle.pem` with `rootca.pem`.

9. Create a Server Certificate PKCS12 file (`cert.p12`) as follows:

   ```
   openssl pkcs12 -export -out cert.p12 -inkey oaa.key -in oaa.pem -chain -
   CAfile bundle.pem
   ```

   When prompted enter and verify the Export Password.

   > **Note:**
   >
   > If your CA does not have a certificate chain replace `bundle.pem` with `rootca.pem`.

> **Note:**
>
> The path to `cert.p12` and `trust.p12` will be used later in the parameters `common.local.sslcert` and `common.local.trustcert` in the `installOAA.properties`.

Generate your own CA and certificates for testing purposes

1. Create a Trusted Certificate PKCS12 file (`trust.p12`) as follows:
   a. On the node where the Management container installation will be run from, create a directory and navigate to that folder, for example:

      ```
      mkdir <workdir>/oaa_ssl
      export WORKDIR=<workdir>
      cd $WORKDIR/oaa_ssl
      ```

   b. Generate a 4096-bit private key for the root Certificate Authority (CA):

      ```
      openssl genrsa -out ca.key 4096
      ```

   c. Create a self-signed root CA certificate (`ca.crt`):

      ```
      openssl req -new -x509 -days 3650 -key ca.key -out ca.crt
      ```

      When prompted enter the details to create your CA. For example:

      ```
      You are about to be asked to enter information that will be
      incorporated
      into your certificate request.
      What you are about to enter is what is called a Distinguished Name or
      a DN.
      There are quite a few fields but you can leave some blank
      For some fields there will be a default value,
      If you enter '.', the field will be left blank.
      -----
      Country Name (2 letter code) [XX]:US
      State or Province Name (full name) []:California
      ```

```
Locality Name (eg, city) [Default City]:Redwood City
Organization Name (eg, company) [Default Company Ltd]:Example
Company
Organizational Unit Name (eg, section) []:Security
Common Name (eg, your name or your server's hostname) []:OAA
Certificate Authority
Email Address []:
```

d. Generate a PKCS12 file for the CA certificate:

```
openssl pkcs12 -export -out trust.p12 -nokeys -in ca.crt
```

When prompted enter and verify the Export Password.

2. Create a Server Certificate PKCS12 file (`cert.p12`) as follows:

   a. Generate a 4096 bit private key (`oaa.key`) for the server certificate:

   ```
   openssl genrsa -out oaa.key 4096
   ```

   b. Create a Certificate Signing Request (`cert.csr`):

   ```
   openssl req -new -key oaa.key -out cert.csr
   ```

   When prompted enter details to create your Certificate Signing Request (CSR). For example:

   ```
   You are about to be asked to enter information that will be
   incorporated
   into your certificate request.
   What you are about to enter is what is called a Distinguished
   Name or a DN.
   There are quite a few fields but you can leave some blank
   For some fields there will be a default value,
   If you enter '.', the field will be left blank.
   -----
   Country Name (2 letter code) [XX]:US
   State or Province Name (full name) []:California
   Locality Name (eg, city) [Default City]:Redwood City
   Organization Name (eg, company) [Default Company Ltd]:Example
   Company
   Organizational Unit Name (eg, section) []:Security
   Common Name (eg, your name or your server's hostname)
   []:oaa.example.com
   Email Address []:

   Please enter the following 'extra' attributes
   to be sent with your certificate request
   A challenge password []:
   An optional company name []:
   ```

c. Generate a certificate from the CSR using the CA created earlier:

```
openssl x509 -req -days 1826 -in cert.csr -CA ca.crt -CAkey ca.key -
set_serial 01 -out oaa.crt
```

d. Generate a PKCS12 file (`cert.p12`) from the private key and server certificate:

```
openssl pkcs12 -export -out cert.p12 -inkey oaa.key -in oaa.crt -
chain -CAfile ca.crt
```

When prompted enter and verify the Export Password.

Note:

The path to `cert.p12` and `trust.p12` will be used later in the parameters `common.local.sslcert` and `common.local.trustcert` in the `installOAA.properties`.

3.2.10 Create a Kubernetes Namespace and Secret

Create a Kubernetes namespace and secret for the OAA and OARM deployment.

1. Create a Kubernetes namespace for the OAA and OARM deployment:

```
kubectl create namespace <namespace>
```

For example:

```
kubectl create namespace oaans
```

Note:

The namespace given will be used later in the parameter `common.kube.namespace=oaans` in the `installOAA.properties`.

2. Create a Kubernetes secret for your Container Image Registry (CIR) in the OAA namespace. This is required so the management container pod can push images to your CIR and so the OAA/OARM deployment can pull images from your CIR.

```
kubectl create secret docker-registry dockersecret --docker-
server=<CONTAINER_REGISTRY> \
 --docker-username='<USER_NAME>' \
--docker-password='<PASSWORD>' \
--docker-email='<EMAIL_ADDRESS>' \
--namespace=<namespace>
```

For example:

```
kubectl create secret docker-registry dockersecret --docker-
server=container-registry.example.com \
--docker-username="user@example.com" \
--docker-password=<PASSWORD> --docker-email=user@example.com \
--namespace=oaans
```

Note:

The secret name `dockersecret` will be used later in the parameter `install.global.imagePullSecrets\[0\].name` in the `installOAA.properties`.

Next Steps:Preparing the Properties file for OAA and OARM Installation

3.3 Downloading Installation Files and Preparing the Management Container

This section provides steps for downloading installation files and preparing the Management Container for OAA and OARM.

The Management Container installation can take place from any node that has access to deploy to the Kubernetes cluster. During installation the Management Container pod will deploy to one of the nodes in the Kubernetes cluster.

1. Download the latest OAA installation Image `<OAA_Image>.zip` from My Oracle Support by referring to the document ID 2723908.1.
2. On the node where the Management container installation will run from, create a `$WORKDIR/oaaimages` directory and copy the `<OAA_Image>.zip` to it:

   ```
   mkdir -p $WORKDIR/oaaimages
   cd $WORKDIR/oaaimages
   cp <download_location>/<OAA_Image>.zip .
   unzip <OAA_Image>.zip
   ```

   This will give you a `<tar_file_name>.tar` file.
3. Navigate to the `$WORKDIR/oaaimages/oaa-install` directory and copy the install template file to `installOAA.properties`:

   ```
   cd $WORKDIR/oaaimages/oaa-install
   cp installOAA.properties.template installOAA.properties
   ```
4. Prepare this `installOAA.properties` as per Preparing the Properties file for OAA and OARM Installation

3.4 Preparing the Properties file for OAA and OARM Installation

You can customize the OAA, OAA-OARM and OARM installation by setting properties in the `installOAA.properties` file. The `installOAA.properties` is used by the Management Container installation script and is copied to `<NFS_CONFIG_PATH>` during the installation of the Management Container pod. The `installOAA.properties` file is later passed as an argument to the `OAA.sh` script when deploying OAA and/or OARM. See Deploying OAA and OARM.

The following sections provide description for the customizations allowed in the `installOAA.properties`.

- Common Deployment Configuration
- Database Configuration
- OAM OAuth Configuration
- Vault configuration
- Helm Chart Configuration
- Optional Configuration
- Ingress Configuration
- Management Container Configuration

3.4.1 Common Deployment Configuration

This section provides details about the common deployment configuration properties that can be set in the `installOAA.properties`.

Table 3-4 Common Deployment Configuration

Properties	Mandatory/Optional	Installation Type	Description
`common.dryrun`	Optional	OAA, OAA-OARM, and OARM	If enabled and set to true, the helm installation will only display generated values and will not actually perform the OAA/OARM installation on the Kubernetes cluster. This is equivalent to `--dry-run --debug` option in the helm command.
`common.deployment.name`	Mandatory	OAA, OAA-OARM, and OARM	Name of the OAA installation. It is unique per kubernetes cluster and namespace when the helm install command is run. The value given must be in lowercase.

Table 3-4 (Cont.) Common Deployment Configuration

Properties	Mandatory/Optional	Installation Type	Description
`common.deployment.overridefile`	Optional	OAA, OAA-OARM, and OARM	Override file for chart parameters override. The helm charts are present in `helmcharts` directory inside the management container. All the parameters defined in `values.yaml` can be overridden by this file, if enabled. The format of this file should be YAML only. A sample `oaaoverride.yaml` file is present in the `~/installsettings` directory inside the management container.
`common.kube.context`	Optional	OAA, OAA-OARM, and OARM	Name of the Kubernetes context to be used. If the context is not provided, the default Kubernetes context is used.
`common.kube.namespace`	Optional	OAA, OAA-OARM, and OARM	The namespace where you want to create the OAA deployment. This should be the namespace created in Create a Kubernetes Namespace and Secret. If the parameter is not set it will deploy to the default namespace.
`common.deployment.sslcert`	Mandatory	OAA, OAA-OARM, and OARM	The server certificate PKCS12 file to be used in the OAA installation. The file name, for example `cert.p12`, is the same file name as the one generated in Generating Server Certificates and Trusted Certificates. The PATH should not change as this is the internal path mapped inside the container. The file is seeded into the vault and downloaded by all OAA microservices

Table 3-4 (Cont.) Common Deployment Configuration

Properties	Mandatory/Optional	Installation Type	Description
`common.deployment.trustcert`	Mandatory	OAA, OAA-OARM, and OARM	The trusted certificate PKCS12 file to be used in the OAA installation. The file name, for example `trust.p12`, is the same file name as the one generated in Generating Server Certificates and Trusted Certificates. The PATH should not change as this is the internal path mapped inside the container. The file is seeded into the vault and downloaded by all OAA microservices
`common.deployment.importtruststore`	Mandatory	OAA, OAA-OARM, and OARM	If this is enabled then the trusted certificate is imported in the JRE truststore.
`common.deployment.keystorepassphrase`	Mandatory	OAA, OAA-OARM, and OARM	Passphrase for the certificate PKCS12 file. This is the passphrase used when creating the keystore in Generating Server Certificates and Trusted Certificates. If you do not specify the value here, you are prompted for the value during installation.
`common.deployment.truststorepassphrase`	Mandatory	OAA, OAA-OARM, and OARM	Passphrase for the trusted certificate PKCS12 file. This is the passphrase used when creating the trusted keystore in Generating Server Certificates and Trusted Certificates If you do not specify the value here you are prompted for the value during installation.

Table 3-4 (Cont.) Common Deployment Configuration

Properties	Mandatory/Optional	Installation Type	Description
`common.deployment.generate.secret`	Mandatory	OAA, OAA-OARM, and OARM	If set to true, the installation generates three symmetric keys and adds them to the `cert.p12` referenced by the parameter `common.deployment.sslcert`. The encryption keys generated are: • spui-enckey - This key is used by the SPUI service for encryption. • aes256_db_key_alias - This key is used for encrypting user runtime information in the database such as users questions/ answers for Knowledge Based Authentication (KBA). • aes256_config_key_alias - This key is for encrypting all the system related configuration. If you create these keys yourself then the value must be set to `false`. To create the keys, run the following command: `keytool -genseckey -alias $keynametouse -keyalg $KEYALGO -keystore $KEYSTORE -storepass $STOREPASS -storetype $STORETYPE -keysize $KEYSIZE`

Table 3-4 (Cont.) Common Deployment Configuration

Properties	Mandatory/Optional	Installation Type	Description

			for example:
			
			`keytool -genseckey -alias spui-enckey -keyalg AES -keystore cert.p12 -storepass <password> -storetype PKCS12 -keysize 256`

`common.deployment.mode`	Mandatory	OAA, OAA-OARM, and OARM	The following values can be set in `installOAA.properties` • `Both` - install OAA and OARM. • `OAA` - install OAA only. • `Risk` - install OARM only.
`common.migration.configkey`	Optional	OAA, OAA-OARM, and OARM	Base64 encoded config key from the transitioning system. If enabled, the value is placed in the vault and used for transitioning of legacy data. Use this only if you transition from Oracle Adaptive Access Manager 11gR2PS3.
`common.migration.dbkey`	Optional	OAA, OAA-OARM, and OARM	Base64 encoded Database key from the transitioning system. If enabled, the value is placed in the vault and used for transitioning of database data. Use this only if you transition from Oracle Adaptive Access Manager 11gR2PS3.

Table 3-4 (Cont.) Common Deployment Configuration

Properties	Mandatory/Optional	Installation Type	Description
`common.oim.integration`	Optional	OAA and OAA-OARM	To integrate with OIM, set the property to true. This also enables the forgot password functionality. Use this only if you transition from Oracle Adaptive Access Manager 11gR2PS3.
`common.deployment.push.apnsjksfile`	Optional	OAA and OAA-OARM	File used when enabling push factor for the Apple Push Notification Service. You need to set this only if you have already configured the JKS file prior to install. Else, you can configure this post installation. The JKS file should be copied to the `<NFS_VAULT_PATH>/ChallengeOMAPUSH/apns/` directory. The value should be set to `/u01/oracle/service/store/oaa/ChallengeOMAPUSH/apns/APNSCertificate.jks`. For more details, see Configuring Oracle Mobile Authenticator Push Notification for iOS.

3.4.2 Database Configuration

This section provides details about the database configuration properties that can be set in the `installOAA.properties`.

Table 3-5 Database Configuration

Properties	Mandatory/Optional	Description
`database.createschema`	Mandatory	Enables creation of the schema during installation. If this is set to `false`, the schema is not created. However, irrespective of this flag, database validation is performed.

Table 3-5 (Cont.) Database Configuration

Properties	Mandatory/Optional	Description
`database.host`	Mandatory	Specify the database hostname or IP address.
`database.port`	Mandatory	Specify the database port..
`database.sysuser`	Mandatory	Specify the `sysdba` user of the database.
`database.syspassword`	Mandatory	Specify the sys password. If you do not specify the value here, you are prompted for value during installation.
`database.schema`	Mandatory	Specify the name of the database schema to be used for installation.
`database.tablespace`	Mandatory	Specify the tablespace name to be used for the installation.
`database.schemapassword`	Mandatory	Specify the schema password. If you do not specify the value here, you are prompted for value during installation.
`database.svc`	Mandatory	Specify the database service name.
`database.name`	Mandatory	Specify the database name. This can be the same as database service name. This parameter is not required if using a RAC database.

Note:

If using a secure connection to an Oracle Database via SSL, then additional configuration steps are required. These steps must be performed after the Management Container is started, and before: Deploying OAA and OARM:

1. Obtain the Oracle Wallet for the Database:
   a. For a standard Oracle database refer to your Database specific documentation for details on how to find the Oracle Database Wallet.
   b. For an Oracle Autonomous Database on Shared Exadata Infrastructure (ATP-S) database follow: Download Client Credentials.
2. Create a `db_wallet` directory in the `<NFS_CONFIG_PATH>` used by the OAA deployment. Copy the wallet file(s) to the `<NFS_CONFIG_PATH>/db_wallet` directory.
3. Enter a bash shell for the OAA management pod:

   ```
   kubectl exec -n <namespace> -ti <oaamgmt-pod> -- /bin/bash
   ```

   For example:

   ```
   kubectl exec -n oaans -ti oaamgmt-oaa-mgmt-7dfccb7cb7-
   lj6sv9 -- /bin/bash
   ```

4. Inside the container set the `TNS_ADMIN` environment variable:

   ```
   export TNS_ADMIN=<NFS_CONFIG_PATH>/db_wallet
   ```

   The `db_wallet` directory must have the correct read and write access privileges to be accessible from inside the container.
5. Deploy OAA as per Deploying OAA and OARM.

3.4.3 OAM OAuth Configuration

This section provides details about the OAM OAuth configuration properties that can be set in the `installOAA.properties`.

Ensure you have followed the prerequisite steps for configuring OAM for OAuth. For details, see Installing and Configuring OAM OAuth .

Table 3-6 OAM OAuth Configuration

Properties	Mandatory/Optional	Description
`oauth.enabled`	Mandatory	OAuth is required if you want to use the OAA Administration User Interface (UI) and OAA User Preferences UI. If access to the UI's is required, you must set this to `true` to enable OAuth in the OAA installation. If you do not want access to the UI's set this to `false`. If you set `oauth.enabled=false` you must also set the following properties to `false`, otherwise the installation fails: • `oauth.createdomain` • `oauth.createresource` • `oauth.createclient` If `oauth.enabled=false` you must also set these parameters to `false` under Optional Configuration: • `install.spui.enabled` • `install.oaa-admin-ui.enabled` • `install.fido.enabled` • `install.oaa-kba.enabled=false`
`oauth.createdomain`	Optional	Creates the OAuth domain. The OAuth domain is required to create OAuth resource and client.
`oauth.createresource`	Optional	Creates the OAuth resource. The OAuth resource is required to create the OAuth client.
`oauth.createclient`	Optional	Creates the OAuth client. The OAuth client is required if `oauth.enabled` is set to `true`.
`oauth.domainname`	Mandatory if `oauth.createdomain` is set to `true`	Specify the OAuth domain name. This must be same as the `<DomainName>` provided in Installing and Configuring OAM OAuth.
`oauth.identityprovider`	Mandatory if `oauth.createdomain` is set to `true`	Specify the identity provider for the OAM OAuth Domain. This is the name of the **User Identity Store** used in OAM.
`oauth.clientname`	Mandatory if `oauth.createclient` is set to `true`	Specify the OAuth client name that will be created during the installation.

Table 3-6 (Cont.) OAM OAuth Configuration

Properties	Mandatory/Optional	Description
`oauth.clientgrants`	Mandatory if `oauth.createclient` is set to `true`	Specify the client grants for the OAuth client. OAuth client must have `CLIENT_CREDENTIALS`, which is used during validation stage to check OAuth status. Values must be: "PASSWORD","CLIENT_CREDENTIALS","JWT_BEARER","REFRESH_TOKEN","AUTHORIZATION_CODE","IMPLICIT".
`oauth.clienttype`	Mandatory if `oauth.createclient` is set to `true`	Specify the OAuth Client Type. OAM OAuth supports the following client types: PUBLIC_CLIENT, CONFIDENTIAL_CLIENT, MOBILE_CLIENT. As OAuth is used for the OAA Administration and User Preference consoles, PUBLIC_CLIENT should be used.
`oauth.clientpassword`	Mandatory if `oauth.enabled=true`	Specify the password that will be used for the OAuth client. The client password must conform to regex `^[a-zA-Z0-9.\-\/+=@_ ]*$` with a maximum length of 500.
`oauth.resourcename`	Mandatory if `oauth.enabled=true`	Specify the OAuth resource name to be created during installation. Also used for validation of the OAuth setup.
`oauth.resourcescope`	Mandatory if `oauth.enabled=true`	Specify the OAuth resource scope to be created during installation. Also used for validation of the OAuth setup.
`oauth.redirecturl`	Mandatory if `oauth.createclient` is set to `true`	Specify the client redirect URL. Post authentication redirecturl is required. This is used for validating configuration of OAuth services in OAM by generating an access token.
`oauth.applicationid`	Mandatory if `oauth.createclient` is set to `true`	Application ID of OAA protected by oauth. The value can be any valid string. It is required to setup runtime integration between OAM and OAA post OAA installation. See Integrating OAA with OAM.

Table 3-6 (Cont.) OAM OAuth Configuration

Properties	Mandatory/Optional	Description
`oauth.adminurl`	Mandatory if `oauth.enabled=true`	Specify the OAuth Administration URL This is the URL of the OAM Administration Server, for example `http://oam.example.com:7001`..
`oauth.basicauthzheader`	Mandatory if `oauth.enabled=true`	Base64 encoded authorization header for the OAM Adminstration Server. The value can be found by executing: `echo -n weblogic:<password> | base64`.
`oauth.identityuri`	Mandatory if `oauth.enabled=true`	URL of the identity server used to retrieve OIDC metadata using `/.well-known/openid-configuration` endpoint. This is the front-end URL of the OAM Managed server providing runtime support for OAuth Services. For example : `http://ohs.example.com:7777`.

3.4.4 Vault configuration

This section provides details about the vault configuration properties that can be set in the `installOAA.properties`.

If you are using OCI vault, you can ignore the properties to be set for file-based vault.

Table 3-7 Vault Configuration

Properties	Description
`vault.deploy.name`	Name to be used in the vault for this deploymemt. If the name is already present in the vault it will be reused.
`vault.create.deploy`	If the value is set to `true`, vault creation is performed. However, if a vault with the name provided in `vault.deploy.name` already exists then vault creation is skipped.
`vault.provider`	Specify if the vault is OCI or file based. Specify one of the following values: • `fks` • `oci`
The following properties are mandatory for OCI-based vault configurations if you have set `vault.provider=oci`. For for information about creating OCI vault, see Managing Vaults. The OCI vault must exist before setting the parameters below.	
`vault.oci.uasoperator`	Specify the Base64 encoded private key of the user with read and write permission on OCI vault.
`vault.oci.tenancyId`	Specify the Base64 encoded OCI ID of the tenancy id.

Table 3-7 (Cont.) Vault Configuration

Properties	Description
`vault.oci.userId`	Specify the Base64 encoded OCID of the user with read and write permission on OCI vault.
`vault.oci.fpId`	Specify the Base64 encoded finger print of the user with read and write permission on OCI vault.
`vault.oci.compartmentId`	Specify the Base64 encoded OCID of the compartment where the vault exists in OCI.
`vault.oci.vaultId`	Specify the Base64 encoded OCID of the vault on OCI.
`vault.oci.keyId`	Specify the Base64 encoded OCID of the master secret key in OCI vault used to encrypt the secrets in the vault.
The following properties are mandatory for file-based vault configurations if you have set `vault.provider=fks`.	
`vault.fks.server`	Specify the NFS server host name or IP address for the `<NFS_VAULT_PATH>`. For more details, see Configuring NFS Volumes.
`vault.fks.path`	Specify the `<NFS_VAULT_PATH>` which will store the file based vault. For more details, see Configuring NFS Volumes.
`vault.fks.key`	Specify a Base64 encoded password for the file based vault. To find the Base64 encoded version of the password use: `echo -n weblogic:<password> | base64`.
`vault.fks.mountpath`	The mount path in the management container and for installed services where the vault exists. The value of this property must be the same as the value passed through the helm chart. Do not change this value: `/u01/oracle/service/store/oaa`.

3.4.5 Helm Chart Configuration

This section provides details about the helm chart configuration properties that can be set in the `installOAA.properties`.

These properties are passed as input to the helm chart during Installation.

Table 3-8 Helm Chart Configuration

Properties	Mandatory/Optional	Description
`install.global.repo`	Mandatory	Specify the Container Image Registry where the OAA container images exists. For more details, see Installing a Container Image Registry (CIR)

Table 3-8 (Cont.) Helm Chart Configuration

Properties	Mandatory/Optional	Description
`install.riskdb.service.type`	Mandatory	You must set the value of this property always to `ExternalName`, as the database is external to the OAA installation.

Table 3-8 (Cont.) Helm Chart Configuration

Properties	Mandatory/Optional	Description
`install.global.imagePullSecrets\[0\].name`	Mandatory	Specify the Kubernetes secret reference that needs to be used while pulling the container images from the protected Container Image Registry. **Note:** This must be set to the Kubernetes secret th

Table 3-8 (Cont.) Helm Chart Configuration

Properties	Mandatory/Optional	Description
		at you set earlier e.g `dockersecret`. For more details, see Creat

Table 3-8 (Cont.) Helm Chart Configuration

Properties	Mandatory/Optional	Description
		eaKubernetesNamespaceandSecret.

Table 3-8 (Cont.) Helm Chart Configuration

Properties	Mandatory/Optional	Description
`install.global.image.tag`	Mandatory	Update the global image tag to the image tag in your Container Image Registry. **Note:** If you copied the `installOAA.properties.te`

Table 3-8 (Cont.) Helm Chart Configuration

Properties	Mandatory/Optional	Description
		mplate to install OAA.properties this tag will be already set.

Table 3-8 (Cont.) Helm Chart Configuration

Properties	Mandatory/Optional	Description
`install.global.oauth.logouturl`	Optional	Specify the logout URL for OAuth protected resource. This is the front-end URL of the OAM Managed server. For example : `http://ohs.example.com:7777/oam/server/logout`. Required only when `oauth.enabled` is set to `true`.
`install.global.uasapikey`	Mandatory	Specify the REST API key to be used used for protecting rest endpoints in OAA microservice.
`install.global.policyapikey`	Mandatory	Specify the REST API key to be used used for protecting REST endpoints in the OAA policy microservice.
`install.global.factorsapikey`	Mandatory	Specify the REST API key to be used for protecting REST endpoints in the OAA factor microservice.
`install.global.riskapikey`	Optional	Specify the REST API key to be used for protecting REST endpoints in the OAA risk microservice. This parameter is mandatory if performing an OAA-OARM installation or OARM only installation.
In case of OCI vault, the following configurations can be overridden if provided for read-only users during helm installation. If the values are not provided in the following properties then the values are picked from Vault Configuration.		
`install.global.vault.mapId`	Optional	For a pre-existing vault you can provide the Base64 mapId. If the property is set then it validates against the deploy information in the vault.
`install.global.vault.oci.uasoperator`	Optional	Specify the Base64 encoded private key of the user with the read-only permission on the vault.
`install.global.vault.oci.tenancyId`	Optional	Specify the Base64 encoded tenancy id from OCI.
`install.global.vault.oci.userId`	Optional	Specify the Base64 encoded user id from OCI.
`install.global.vault.oci.fpId`	Optional	Specify the Base64 encoded finger print id of the user from the OCI.

3.4.6 Optional Configuration

This section provides details about the optional configuration properties that can be set in the `installOAA.properties`.

Properties	Mandatory/Optional	Description
`install.global.ingress.enabled`	Optional	This property is used to indicate if ingress is to be enabled for the deployment. If the value is set to true, the ingress resource in the Kubernetes cluster for the deployment will be generated. If a pure NodePort based deployment is required, the value should be set to false.
`install.global.ingress.runtime.host`	Optional	You can specify the Host name to be used for ingress definition for the runtime host. If the value for the property is missing, ingress definition is created using '*' host. The runtime host is used for accessing runtime services including all factors, oaa, spui and risk.
`install.global.ingress.admin.host`	Optional	You can specify the Host name to be used for ingress definition for the admin host. If the value for the property is missing, ingress definition is created using '*' host. The admin host is used for accessing admin, policy and risk-cc services.

`install.global.dbhost`
`install.global.dbport`
`install.global.dscredentials`
`install.global.dbservicename`
	Optional	These properties are related to the database. If the property is not specified here, the values provided in the Database Configuration are used.

`install.global.oauth.oidcidentityuri`
`install.global.oauth.oidcaudience`
`install.global.oauth.oidcclientid`
	Optional	The following properties are related to OAuth. If they are not specified here, the values provided in the OAuth Configuration are used.

Properties	Mandatory/Optional	Description
`install.global.serviceurl`	Optional	If load balancer/ingress url is present, then configure the url here. All UI services will be behind this load balancer/ ingress. In case ingress installation is set to true, the appropriate service url will be fetched after ingress installation and will be used as service url. If `install.global.serviceurl` is provided, the service url from this property will have higher priority and override the original value.
`install.oaa-admin-ui.serviceurl`	Optional	Service URL of oaa admin, if different from `install.global.serviceurl`.
`install.spui.enabled=false` `install.fido.enabled=false` `install.oaa-admin-ui.enabled=false` `install.oaa-kba.enabled=false`	Optional	If `oauth.enabled=false` the OAA Admin console (`oaa-admin-ui`), User Preferences console (`spui`) , and FIDO (`fido`) and KBA (`oaa-kba`) factors cannot be used. If `oauth.enabled=false` you must uncomment these properties. When `common.deployment.mode=Risk` the following service are not deployed: fido, push, yotp, email ,sms, totp and kba.
`install.totp.enabled=false` `install.push.enabled=false` `install.sms.enabled=false` `install.yotp.enabled=false` `install.email.enabled=false`		Authentication factor services are enabled by default. To disable them uncomment the lines. When `common.deployment.mode=Risk` the following service are not deployed: fido, push, yotp, email ,sms, totp and kba.

Properties	Mandatory/Optional	Description

`install.service.type=NodePort`
`install.oaa-admin-ui.service.type=NodePort`
`install.oaa-policy.service.type=NodePort`
`install.spui.service.type=NodePort`
`install.totp.service.type=NodePort`
`install.fido.service.type=NodePort`
`install.push.service.type=NodePort`
`install.email.service.type=NodePort`
`install.sms.service.type=NodePort`
`install.yotp.service.type=NodePort`
`install.risk.service.type=NodePort`
`install.oaa-kba.service.type=NodePort`
`install.risk.riskcc.service.type=NodePort`
	Optional	Default service type for services is NodePort.
		When deployment mode is Risk the following service are not deployed : fido, push, yotp, email ,sms, totp and kba.
		If `install.global.ingress.enabled=true` all these parameters should be commented out.

For details on installing using ingress, see: Installing OAA and OARM Using NGINX Ingress

3.4.7 Ingress Configuration

This section provides details about the Ingress configuration properties that can be set in the `installOAA.properties`.

Table 3-9 Ingress Configuration

Properties	Mandatory/Optional	Description
`ingress.install`	Mandatory	Set value to `true` if you want the OAA or OARM installation to install an ingress controller for you. Set to `false` if you do not want to install the ingress controller. If this is set to `true` then `install.global.ingress.enabled=true` must also be set in Optional Configuration.
`ingress.namespace`	Mandatory if ingress.install=true	The Kubernetes namespace which will be used to install ingress. The install will create this namespace in Kubernetes. For example, `ingress-nginx`.
`ingress.admissions.name=ingress-nginx-controller-admission`	Optional if ingress.install=true	The name of the Admissions controller. The Admissions controller can be installed separately. If Ingress admissions name is not present, the `controller.admissionWebhooks.enabled` will be set to `false` in the NGINX ingress chart.
`ingress.class.name=ingress-nginx-class`	Mandatory if ingress.install=true	Ingress class name that needs to be used for the installation. It must not be an existing class name.
`ingress.service.type`	Mandatory if ingress.install=true	Set the value to `NodePort` if using a bare metal Kubernetes cluster. The ingress controller will listen on one of the nodes of the cluster on a dynamically assigned port. Set the value to `LoadBalancer` if you are using a Managed Service for your Kubernetes cluster, for example Oracle Kubernetes Engine (OKE) on Oracle Cloud Infrastructure (OCI). This instructs the Managed Service to setup a Load Balancer to direct traffic to the NGINX ingress.
`ingress.install.releaseNameOverride=base`	Optional if ingress.install=true	Anything starting with `ingress.install` can be additionally supplied to set the ingress chart value.

For details on installing using ingress, see: Installing OAA and OARM Using NGINX Ingress

3.4.8 Management Container Configuration

This section provides details about the Management Container configuration properties that can be set in the `installOAA.properties`.

Table 3-10 Management Configuration

Properties	Mandatory/Optional	Description
`install.mount.config.path`	Mandatory	Set the value of `<NFS_CONFIG_PATH>` to the NFS mount path for the configuration.
`install.mount.config.server`	Mandatory	The IP address of the NFS server for the `<NFS_CONFIG_PATH>`.
`install.mount.creds.path`	Mandatory	Set the value of `<NFS_CREDS_PATH>` to the NFS mount path for the credentials.
`install.mount.creds.server`	Mandatory	The IP address of the NFS server for the `<NFS_CREDS_PATH>`.
`install.mount.logs.path`	Mandatory	Set the value of `<NFS_LOGS_PATH>` to the NFS mount path for the logs.
`install.mount.logs.server`	Mandatory	The IP address of the NFS server for the `<NFS_LOGS_PATH>` .
`install.mgmt.release.name`	Optional	Name of the OAA management container installation used when the helm install command is run. If not set you will be prompted for the name during the installation. The value given must be in lowercase.
`install.kube.creds`	Optional	Set the value to the local PATH where `kubeconfig` resides. If not set the management container will use `$KUBECONFIG` or `~/.kube/config` for Kubernetes credentials.
`common.local.sslcert`	Mandatory	Set the value to the local PATH where the server certificate PKCS12 file (`cert.p12`) resides.
`common.local.trustcert`	Mandatory	Set the value to the local PATH where the trusted certificate PKCS12 file (`trust.p12`) resides.

For details on NFS mounts, see: Configuring NFS Volumes

For details on the PKCS12 files, see: Generating Server Certificates and Trusted Certificates

3.5 Running the Management Container

Run the `installManagementContainer.sh` script to create the Management Container.

1. On the node where you downloaded the installation files, navigate to the `$WORKDIR/oaaimages/oaa-install` directory:

```
cd $WORKDIR/oaaimages/oaa-install
```

2. Run the `installManagementContainer.sh` script with arguments. For example:

```
./installManagementContainer.sh -t ./<oaa-image>.tar
```

A full list of arguments for `installManagementContainer.sh` is shown in the table below:

Command line argument	Mandatory	Description
`-t`	No	Path to the OAA image tar file. **Usage**: • If not provided pull, tag, and push to the Container Image Registry will not be performed. • If pull, tag, and push is required the path to the image `<oaa-image.tar>` must be provided, for example: `-t ./oaa-latest.tar`. • The install script will first attempt to use podman to perform this task, and if not found will use Docker if available. If neither podman nor Docker are available the script will exit.
`-c`	No	Path to OAA management helm chart. If not provided the script will use `./charts/oaa-mgmt` as the path.
`-d`	No	Perform a helm dry-run of the installation.
`-f`	No	Path to `installOAA.properties`. If not provided `./installOAA.properties` will be used.
`-v`	No	Runs the script in verbose mode.

Command line argument	Mandatory	Description
`-p`	No	Set http/https proxies in the OAA management container's environment. By default the proxies will not be set. If specified the script will use its environment to find the proxy configuration to use.
`-e`	No	Add entries to OAA management container's `/etc/hosts`. By default entries are not added. If specified the script will prompt for the information.
`-n`	No	Do not prompt By default the script will prompt for the information it needs to install the OAA management chart and before proceeding from one stage to the next of the install. If this option is set the script will not prompt for missing information or between stages. If required information is missing it will exit in error instead.
`-u`	No	Perform an update instead of an install. By default the script will determine whether to perform and install or an upgrade by looking for the helm chart previously installed.

As the install progresses you will be prompted to answer various questions and perform certain tasks. The table below outlines some of the questions or tasks you may be asked to answer or perform:

Output	Action

`Use 'podman login' to login into your private registry if you have not done so previously.Login successful? [Y/N]:`
 	
Note:
If using Docker the above will show `docker login`.
	If your private Container Image Registry (CIR) where you store images requires a login, use `podman login` or `docker login` to log into the CIR and enter your credentials when prompted:
	
	`podman login <container-registry.example.com>`
	
	or:
	
	`docker login <container-registry.example.com>`

Output	Action
`Would you like to specify a kube context (otherwise 'kubernetes-admin@kubernetes' will be used)? [Y/N]:`	If you have multiple kube contexts in your cluster you can choose which context to use. If you select "Y" you must type the context you wish to use. If you wish to use the default context chosen, or only have one context in your kube config, choose "N".

Note:

The table above does not include an exhaustive list of all the prompts you will see during the install as most are self explanatory.

Once the Management Container installation is complete you will see output similar to the following:

```
NAME: oaamgmt
LAST DEPLOYED: <DATE>
STATUS: deployed
REVISION: 1
TEST SUITE: None
Waiting 15 secs for OAA mgmt deployment to start...
Executing 'kubectl get pods oaamgmt-oaa-mgmt-7dfccb7cb7-lj6sv -n oaans'...
NAME                                    READY   STATUS
RESTARTS   AGE
oaamgmt-oaa-mgmt-7dfccb7cb7-lj6sv   0/1     ContainerCreating
0          15s
Waiting 15 secs for OAA mgmt deployment to run...
Executing 'kubectl get pods oaamgmt-oaa-mgmt-7dfccb7cb7-lj6sv -n oaans'...
NAME                                    READY   STATUS
RESTARTS   AGE
oaamgmt-oaa-mgmt-7dfccb7cb7-lj6sv   0/1     ContainerCreating
0          30s
Waiting 15 secs for OAA mgmt deployment to run...
Executing 'kubectl get pods oaamgmt-oaa-mgmt-7dfccb7cb7-lj6sv -n oaans'...
NAME                                    READY   STATUS
RESTARTS   AGE
oaamgmt-oaa-mgmt-7dfccb7cb7-lj6sv   0/1     ContainerCreating
0          46s
Waiting 15 secs for OAA mgmt deployment to run...
Copying OAA properties file to oaans/oaamgmt-oaa-mgmt-7dfccb7cb7-
lj6sv:/u01/oracle/scripts/settings
Generating kube config for OAA mgmt pod 'oaans/oaamgmt-oaa-mgmt'.
Using service account 'oaans/oaamgmt-oaa-mgmt'.
Using token name 'oaamgmt-oaa-mgmt-token-5m88n'.
Using cluster URL 'https://<URL>'.
Cluster "oaa-cluster" set.
User "oaa-service-account" set.
Context "kubernetes-admin@kubernetes" created.
Switched to context "kubernetes-admin@kubernetes".
Copying OAA kube config files to oaans/oaamgmt-oaa-mgmt-7dfccb7cb7-
lj6sv:/u01/oracle/scripts/creds...
```

```
Using helm config '/home/opc/.config/helm/repositories.yaml'.
Copying helm config to oaans/oaamgmt-oaa-mgmt-7dfccb7cb7-lj6sv:/u01/
oracle/scripts/creds/helmconfig
Copying certificates to oaans/oaamgmt-oaa-mgmt-7dfccb7cb7-
lj6sv:/u01/oracle/scripts/creds
Use command 'kubectl exec -n oaans -ti oaamgmt-oaa-mgmt-7dfccb7cb7-
lj6sv -- /bin/bash' to get a shell to the OAA mgmt pod.
From pod shell, use command 'kubectl get pods' to verify
communication with the cluster.
Continue OAA installation from the OAA mgmt pod.
OAA management installation complete.
```

3. As per the output connect to the OAA management pod, for example:

   ```
   kubectl exec -n oaans -ti oaamgmt-oaa-mgmt-7dfccb7cb7-lj6sv9
   -- /bin/bash
   ```

   This will take you into a bash shell inside the OAA management pod:

   ```
   [oracle@oaamgmt-oaa-mgmt-7dfccb7cb7-lj6sv /]$
   ```

Next Steps: Deploying OAA and OARM

3.6 Deploying OAA and OARM

Before deploying OAA and OARM make sure you have followed Running the Management Container.

Deploying OAA, OAA-OARM, or OARM

1. Enter a bash shell for the OAA management pod if not already inside one:

   ```
   kubectl exec -n <namespace> -ti <oaamgmt-pod> -- /bin/bash
   ```

   For example:

   ```
   kubectl exec -n oaans -ti oaamgmt-oaa-mgmt-7dfccb7cb7-lj6sv9
   -- /bin/bash
   ```

2. Inside the OAA management pod bash shell, deploy OAA, OAA-OARM, or OARM by running the `OAA.sh` script:

   ```
   cd ~
   ./OAA.sh -f installOAA.properties
   ```

   > **Note:**
   >
   > This will use the `installOAA.properties` in the `<NFS_CONFIG_PATH>`.

Next Steps: If the install is successful you will see output similar to Printing Deployment Details.

3.7 Printing Deployment Details

After the deployment completes, the following information is printed on the console.

> **Note:**
>
> In cases where you install the ingress controller that ships with OAA/OARM, the host and port is set to the worker node where the controller gets installed. In cases where you are using your own ingress controller, assuming a basic setup, the host and port is set to the value of the property `install.global.serviceurl` in `installOAA.properties`.
> In cases where ingress is disabled, the host and NodePort of the worker nodes are printed.

```
###########################OAA Deployment Details:
START###################################
OAAService=https://oaainstall-host/oaa/runtime
AdminUrl=https://oaainstall-host/oaa-admin
PolicyUrl=https://oaainstall-host/oaa-policy
SpuiUrl=https://oaainstall-host/oaa/rui
Email=https://oaainstall-host/oaa-email-factor
Push=https://oaainstall-host/oaa-push-factor
Fido=https://oaainstall-host/fido
SMS=https://oaainstall-host/oaa-sms-factor
TOTP=https://oaainstall-host/oaa-totp-factor
YOTP=https://oaainstall-host/oaa-yotp-factor
KBA=https://oaainstall-host/oaa-kba
RELEASENAME=oaainstall
# Key below is Base64 encoded API key
oaaapikey=YXBpa2V5dG9iZXNldGR1cmluZ2luc3RhbGxhdGlvbgo=
# Key below is Base64 encoded Policy API key
oaapolicyapikey=cG9sYXBpa2V5dG9iZXNldGR1cmluZ2luc3RhbGxhdGlvbgo=
# Key below is Base64 encoded Factor API key
oaafactorapikey=ZmFjdG9yYXBpa2V5dG9iZXNldGR1cmluZ2luc3RhbGxhdGlvbgo=
###########################Deployment Details: END###################################
```

For OAA-OARM and OARM installations, the Risk Deployment Details are also printed on the console:

```
###########################Risk Deployment Details:
START###################################
AdminUrl=https://oaainstall-host/oaa-admin
PolicyUrl=https://oaainstall-host/oaa-policy
RISK=https://oaainstall-host/risk-analyzer
RISKCC=https://oaainstall-host/risk-cc
RELEASENAME=riskinstall
# Key below is Base64 encoded Policy API key
oaapolicyapikey=cG9sYXBpa2V5dG9iZXNldGR1cmluZ2luc3RhbGxhdGlvbgo=
# Key below is Base64 encoded Factor API key
riskfactorapikey=cmlza2ZhY3RvcmFwaWtleXRvYmVzZXRkdXJpbmdpbnN0YWxsYXRpb24K
```

```
###########################Deployment Details:
END###################################
```

If you ever need to reprint the deployment information:

1. Enter a bash shell for the OAA management pod if not already inside one:

   ```
   kubectl exec -n <namespace> -ti <oaamgmt-pod> -- /bin/bash
   ```

   For example:

   ```
   kubectl exec -n oaans -ti oaamgmt-oaa-mgmt-7d7597c694-vn4ds -- /bin/
   bash
   ```

2. Run `printOAADetails.sh` to print the deployment details:

   ```
   cd ~/scripts
   ./printOAADetails.sh -f settings/installOAA.properties
   ```

   > **Note:**
   >
   > This will use the `installOAA.properties` in the `<NFS_CONFIG_PATH>`.

Based on the information printed for the deployment, the consoles can be accessed as follows:

Console	Print Details Reference *	URL	Username	Password
OAA/OARM Administration	`AdminUrl`	`https://<hostname.domain>:<port>/oaa-admin`	`oaadmin`	`<password>` set in OAM OAuth identity store.
OAA User Preferences	`SpuiUrl`	`https://<hostname.domain>:<port>/oaa/rui`	Username from OAM OAuth identity store.	`<password>` set in OAM OAuth identity store.

** Throughout this documentation the value in the **Print Details Reference** column is used to denote the URL to use. For example: "Launch a browser and access the `<AdminURL>`", refers to accessing the corresponding URL `https://<hostname.domain>:<port>/oaa-admin` shown.*

Based on the information printed for the deployment, the REST API endpoint information is as follows:

REST API	Print Details Reference **	URL	Username **	Password **
OAA/OARM Admin	`PolicyUrl`	`https://<hostname.domain>:<port>/oaa-policy`	`<RELEASENAME>-oaa-policy`	`<Base64Decoded(oaapolicyapikey)>`
OAA Runtime	`OAAService`	`https://<hostname.domain>:<port>/oaa/runtime`	`<RELEASENAME>-oaa`	`<Base64Decoded(oaaapikey)>`
Risk	`RISK`	`https://<hostname.domain>:<port>/risk-analyzer`	`<RELEASENAME>-risk`	`<Base64Decoded(riskfactorapikey)>`
Risk Customer Care	`RISKCC`	`https://<hostname.domain>:<port>/risk-cc`	`<RELEASENAME>-risk-cc`	`<Base64Decoded(riskfactorapikey)>`
KBA	`KBA`	`https://<hostname.domain>:<port>/oaa-kba`	`<RELEASENAME>_OAA_KBA`	`<Base64Decoded{oaafactorsapikey}>`

*** Throughout this documentation, when REST API examples are given, the value in the **Print Details Reference** column is used to denote the URL to use, and the values in the **Username** and **Password** columns represent the username and password to use.*

For example:

```
curl --location -g --request POST '<OAAService>/preferences/v1' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic <Base64Encoded(<username>:<password>)>'
etc...
```

`<OAAService>` refers to accessing the corresponding URL `https://<hostname.domain>:<port>/oaa/runtime`, and <username> refers to `<RELEASENAME>-oaa`*, and <password> refers to* `<Base64Decoded(oaaapikey)>`.

For more information about using the above details to access REST APIs, see:

- OAA Admin API
- OAA RuntimeAPI
- OARM Risk API
- OARM Risk Customer Care API

3.8 Post Installation Steps

Topics

- Post-Installation Steps for NodePort

- Post Installation Steps for all installations
- Post Installation Steps for OAA-OARM and OARM installs

3.8.1 Post Installation Steps for all installations

Follow these post installation steps for all installation types; OAA only , OAA-OARM and OARM only.

Import the policy snapshot by performing the following steps:

1. Obtain the latest snapshot policy file from My Oracle Support by referring to the document ID 2723908.1.

   Note:

   If the latest snapshot file is not available, use the `/u01/oracle/scripts/oarm-12.2.1.4.1-base-snapshot.zip` file inside the management container.

2. Copy the latest snapshot file to the `<NFS_CONFIG_PATH>`.
3. Edit the `<NFS_CONFIG_PATH>/installOAA.properties` and add the following parameters to the bottom of the **Common Deployment** configuration:

   ```
   common.deployment.import.snapshot=true
   common.deployment.import.snapshot.file=/u01/oracle/scripts/settings/
   <latest_snapshot>.zip
   ```

   Note:

   If not using the latest snapshot file set: `common.deployment.import.snapshot.file=/u01/oracle/scripts/settings/oarm-12.2.1.4.1-base-snapshot.zip`

4. Enter a bash shell for the OAA management pod if not already inside one:

   ```
   kubectl exec -n <namespace> -ti <oaamgmt-pod> -- /bin/bash
   ```

   For example:

   ```
   kubectl exec -n oaans -ti oaamgmt-oaa-mgmt-7dfccb7cb7-lj6sv9
   -- /bin/bash
   ```

5. Run the following command inside the bash shell to import the policy snapshot:

   ```
   cd ~/scripts
   ./importPolicySnapshot.sh -f settings/installOAA.properties
   ```

Note:

This will use the `installOAA.properties` in the `<NFS_CONFIG_PATH>`.

The snapshot will import and if successful you will see the message `Successfully applied snapshot: 10001` or similar.

Exit the bash shell.

6. Edit the `<NFS_CONFIG_PATH>/installOAA.properties` and change the following parameter at the bottom of the **Common Deployment** configuration:

```
common.deployment.import.snapshot=false
```

Note:

This is an important step to avoid overwriting and erasing policies during a future upgrade.

3.8.2 Post Installation Steps for OAA-OARM and OARM installs

Follow these post installation steps for OAA-OARM and OARM only installations.

Set oaa.browser.cookie.domain and oaa.risk.integration.postauth.cp

Note:

The steps below are only applicable to OAA-OARM installations.

The properry `oaa.browser.cookie.domain` must be set to the OAA host domain in order to collect the device cookie. For example, if the OAA is accessible on `https://oaa.example.com`, then set the value to `oaa.example.com`.

The property `oaa.risk.integration.postauth.cp` must be set to `postauth` to invoke risk rules for usecases such as Risky IP, Geo-velocity, and Geo-location.

1. Set the properties as follows:
Use the `<PolicyUrl>/policy/config/property/v1` REST API to set the properties. For example:

```
curl --location -g --request PUT '<PolicyUrl>/policy/config/property/v1' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic <Base64Encoded(<username>:<password>)>' \
--data '[
    {
        "name": "oaa.browser.cookie.domain",
        "value": "<host.domain>"
    },
    {
        "name": "oaa.risk.integration.postauth.cp",
        "value": "postauth"
    }
]'
```

> **Note:**
>
> In this case remove `/oaa-policy` from the `<PolicyUrl>`, for example use `https://<host>:<port>/policy/config/property/v1` not `https://<host>:<port>/oaa-policy/policy/config/property/v1`

For details about finding the `PolicyUrl` and authenticating, see OAA Admin API.

For details about the Configuration Properties REST Endpoint, see Configuration Properties REST Endpoints.

3.8.3 Post-Installation Steps for NodePort

For all installation types, if you are using nodeport as oppose to ingress, you must update the OAuth client with the relevant redirect URLs. Perform the following steps:

1. Find the URLs for the `spui`, `oaa-admin-ui`, and `fido` pods as described in Printing Deployment Details, for example:

   ```
   AdminUrl=https://worker1.example.com:32701/oaa-admin
   SpuiUrl=https://worker1.example.com:32721/oaa/rui
   Fido=https://worker1.example.com:30414/fido
   ```

   > **Note:**
   >
   > For OARM only installation, you only need to find the URL for the `oaa-admin-ui` pod.

2. Encode the OAM administrator user and its password by using the command:

   ```
   echo -n <username>:<password> | base64
   ```

   For example:

   ```
   echo -n weblogic:<password> | base64
   ```

   This value should be used for `<ENCODED_OAMADMIN>` in the examples below.

3. For OAA and OAA-OARM installation, update the OAuth Client using REST APIs as follows:

   ```
   curl --location --request PUT 'http://<OAuth_Host>:<OAuth_Port>/oam/
   services/rest/ssa/api/v1/oauthpolicyadmin/client?name=OAAClient' \
   --header 'Content-Type: application/json' \
   ```

```
--header 'Authorization: Basic <ENCODED_OAMADMIN>' \
--data '{
    "id": "OAAClient",
    "clientType": "PUBLIC_CLIENT",
    "idDomain": "OAADomain",
    "name": "OAAClient",
    "redirectURIs": [
        {
            "url": "https://worker1.example.com:32701/oaa/rui",
            "isHttps": true
        },
        {
            "url": "https://worker1.example.com:32701/oaa/rui/oidc/
redirect",
            "isHttps": true
        },
        {
            "url": "https://worker1.example.com:32721/oaa-admin",
            "isHttps": true
        },
        {
            "url": "https://worker1.example.com:32721/oaa-admin/oidc/
redirect",
            "isHttps": true
        },
        {
            "url": "https://worker1.example.com:30414/fido",
            "isHttps": true
        },
        {
            "url": "https://worker1.example.com:30414/fido/oidc/redirect",
            "isHttps": true
        }
    ]
}'
```

Note: For details about the REST API see, REST API for OAuth in Oracle Access Manager

For OARM only installation, update the OAuth Client as follows:

```
curl --location --request PUT 'http://<OAuth_Host>:<OAuth_Port>/oam/
services/rest/ssa/api/v1/oauthpolicyadmin/client?name=OAAClient' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic <ENCODED_OAMADMIN>' \
--data '{
    "id": "OAAClient",
    "clientType": "PUBLIC_CLIENT",
    "idDomain": "OAADomain",
    "name": "OAAClient",
    "redirectURIs": [
      {
            "url": "https://worker1.example.com:32721/oaa-admin",
            "isHttps": true
        },
        {
            "url": "https://worker1.example.com:32721/oaa-admin/
oidc/redirect",
            "isHttps": true
        }
    ]
}'
```

3.9 Troubleshooting OAA and OARM Installation

This section provides troubleshooting tips for installing OAA and OARM.

Podman issues during OAA Management Container installation

- Podman fails to load the OAA images in the tar file due to image or file format errors. For example:

  ```
  Storing signatures
  Getting image source signatures
  Copying blob 01092b6ac97d skipped: already exists
  Copying blob dba9a6800748 skipped: already exists
  Copying blob bae273a35c58 skipped: already exists
  Copying blob 7f4b55b885b0 skipped: already exists
  Copying blob 93e8a0807a49 skipped: already exists
  Copying blob fa5885774604 skipped: already exists
  Copying blob 3b8528487f10 skipped: already exists
  Copying blob 3a1c2e3e35f4 [==========================>-----------]
  213.8MiB / 298.1MiB
  Copying blob 6d31843e131e [=================================>----]
  210.5MiB / 236.5MiB
  Copying blob f35b9630ef38 [===========>--------------------------]
  213.8MiB / 672.2MiB
  Copying blob ef894c2768e3 done
  Copying blob 846fd069f886 [==========>---------------------------]
  197.7MiB / 672.2MiB
  Copying blob 257c48b76c82 done
  Error: payload does not match any of the supported image formats
  (oci, oci-archive, dir, docker-archive)
  ```

  This may happen because of lack of free space in the root partition of the installation host (podman stores temporary files under `/var/tmp`), or because the podman version is not 3.3.0 or later. If this error occurs, remove all files under `/var/tmp` before retrying the installation once the issues have been addressed.

- Podman fails to load the OAA images in the tar file due to permissions issues. For example:

  ```
  Using image release files ./releaseimages.txt and ./
  nonreleaseimages.txt...
  tee: ./oaainstall-tmp/run.log: Permission denied
  Using install settings from ./installOAA.properties.
  tee: ./oaainstall-tmp/run.log: Permission denied
  Checking kubectl client version...
  ```

```
WARNING: version difference between client (1.23) and server (1.21)
exceeds
the supported minor version skew of +/-1
tee: ./oaainstall-tmp/run.log: Permission denied
kubectl version required major:1 minor:18, version detected major:1
minor:23
tee: ./oaainstall-tmp/run.log: Permission denied
```

This may happen if you extract the zip file as one user and run `installManagementContainer.sh` as a different user who doesn't have permissions. In this situation remove the `$WORKDIR/oaaimages/oaa-install/oaainstall-tmp` directory and retry the install with the same user who extracted the zip file.

- Podman failed to load the OAA images in the previous attempt to install and now it won't pull/tag/push of all required images. In this situation remove the `$WORKDIR/oaaimages/oaa-install/oaainstall-tmp` directory and retry.

OAA Management chart installation failure

If the OAA management chart installation fails with the following:

```
Executing 'helm install ...  oaamgmt charts/oaa-mgmt'.
Continue? [Y/N]:
y
Error: unable to build kubernetes objects from release manifest: error
validating "": error validating data:
ValidationError(Deployment.spec.template.spec.containers[0]): unknown field
"volumMounts" in io.k8s.api.core.v1.Container
```

it is likely that the manifest files for the OAA management chart got corrupted. Copy `installOAA.properties`, `cert.p12`, and `trust.p12` to a safe location, remove the install directory `$WORKDIR/oaaimages/oaa-install`, extract the `<OAA_Image>.zip` and restart the installation.

Installation script times out waiting for OAA Management Container pod to start

If you see the following error:

```
NAME                                     READY   STATUS
RESTARTS   AGE
oaamgmt-oaa-mgmt-74c9ff789d-wq82h   0/1     ContainerCreating   0
2m3s
Waiting 15 secs for OAA mgmt deployment to run...
Executing 'kubectl get pods oaamgmt-oaa-mgmt-74c9ff789d-wq82h -n oaans'...
NAME                                     READY   STATUS
RESTARTS   AGE
oaamgmt-oaa-mgmt-74c9ff789d-wq82h   0/1     ContainerCreating   0
2m18s
Waiting 15 secs for OAA mgmt deployment to run...
...
OAA mgmt pod is not running after 450 secs, cannot proceed with install.
Critical error, exiting. Check ./oaainstall-tmp/run.log for additional
information.
```

then run the following commands to get additional information:

```
$ kubectl get pods -n oaans
$ kubectl describe pod oaamgmt-<pod> -n oaans
```

- In case of NFS errors, verify that the NFS volume information in `installOAA.properties` is correct. In this situation `kubectl describe` will show the following:

  ```
  Output: mount.nfs: mounting <ipaddress>:/scratch/oaa/scripts-creds
  failed, reason given by server: No such file or directory
    Warning  FailedMount  15s  kubelet, <ipaddress>  Unable to attach
  or mount volumes: unmounted volumes=[oaamgmt-oaa-mgmt-configpv
  oaamgmt-oaa-mgmt-credpv oaamgmt-oaa-mgmt-logpv], unattached
  volumes=[oaamgmt-oaa-mgmt-configpv oaamgmt-oaa-mgmt-credpv oaamgmt-
  oaa-mgmt-logpv oaamgmt-oaa-mgmt-vaultpv default-token-rsh62]: timed
  out waiting for the condition
  ```

- In case of image pull errors verify that the image pull secret (`dockersecret`) was created correctly, and that the properties `install.global.repo`, `install.global.image.tag, and install.global.imagePullSecrets\[0\].name` in `installOAA.properties` are correct. In this situation `kubectl describe pod` will show the following:

  ```
  Warning  Failed     21s (x3 over 61s)  kubelet, <ipaddress>  Error:
  ErrImagePull
  Normal   BackOff    7s (x3 over 60s)   kubelet, <ipaddress>  Back-
  off pulling image "container-registry.example.com/oracle/shared/oaa-
  mgmt:<tag>"
  Warning  Failed     7s (x3 over 60s)   kubelet, <ipaddress>  Error:
  ImagePullBackOff
  ```

- In case of timeouts with no apparent error it may be possible that the cluster took too long to download the OAA management image. In this case the management pod will eventually start but the installation will abort. If this happens, delete the OAA management helm release using `helm delete oaamgmt -n oaans` and rerun the installation script.

General failures during OAA.sh

If the `OAA.sh` deployment fails at any stage during the install you can generally fix the issue and rerun `OAA.sh`. The install performs a number of checks against the Database, OAuth, and Vault. If re-running the `OAA.sh` fails at these checks because the Database schema, OAuth configuration, or Vault already exists, then set these properties in the `installOAA.properties` before trying the `OAA.sh` again:

- If Database schema is already present:
  - `database.createschema=false`
- If OAuth configuration is already present:
  - `oauth.createdomain=false`
  - `oauth.createresource=false`

  - `oauth.createclient=false`
- If Vault configuration is present:
  - `vault.create.deploy=false`

OAuth creation fails during OAA.sh

During the installation, the OAuth domain, client, and resource server are created. If they fail, check if the parameters for OAuth are correct. See Installing and Configuring OAM OAuth.

OAuth check fails during OAA.sh

This occurs if the `httpd.conf` and `mod_wl_ohs.conf` files are not updated. To update the values, see Installing and Configuring OAM OAuth.

During OAA.sh installation fails because of pods in Container Creating status

Run the following command to check the logs. For example:

```
kubectl logs oaainstall-email-6fd7c9b9dd-lr5lm
```

If the logs do not provide the required details about the error, run the `describe pod` command. For example:

```
kubectl describe pod oaainstall-email-6fd7c9b9dd-lr5lm
```

During OAA.sh pods fail to start and show CrashLoopBackOff

Run the `kubectl logs <pod>` command against the pods showing the error. The following may be one of the reasons for the error:

Pods were not able to connect to `http://www.example.oracle.com:7791/.well-known/openid-configuration` because the `PathTrim` and `PathPrepend` in the `mod_wl_ohs.conf` for that entry were not updated. See Installing and Configuring OAM OAuth.

OAA.sh installation timed out but pods show as running

If the OAA installation timed out but the OAA pods show no errors and eventually end up in running state, it is possible that the cluster took too long to download the OAA images. In this case the OAA pods will eventually start but the installation will not complete. If this happens, clean up the installation and rerun the installation script.

kubectl reports "Unable to connect to the server: net/http: TLS handshake timeout"

Possible causes are:

- Proxies are defined in the environment and the `no_proxy`" environment variable does not include the cluster nodes. To resolve the issue the cluster node IPs or hostnames must be added to the `no_proxy` environment variable.
- The kube config file `~/.kube/config` or `/etc/kubernetes/admin.conf` is not valid.

Unable to delete the OAA domain from OAuth during cleanup

List all clients and resources within the domain and delete each one of them before deleting the domain:

1. Encode the OAM administrator user and its password by using the command:

   ```
   echo -n <username>:<password> | base64
   ```

   For example:

   ```
   echo -n weblogic:<password> | base64
   ```

   This value should be used for `<ENCODED_OAMADMIN>` in the example below.

2. Run the following:

   ```
   $ curl --location --request DELETE 'http://
   <OAuth_Host>:<OAuth_port>/oam/services/rest/ssa/api/v1/
   oauthpolicyadmin/oauthidentitydomain?name=OAADomain' \
   --header 'Authorization: Basic <ENCODED_OAMADMIN>'
   OAuth Identity Domain is not empty. Kindly remove (resource/client)
   entities from identity domain
   $ curl --location --request GET 'http://
   <OAuth_Host>:<OAuth_port>/oam/services/rest/ssa/api/v1/
   oauthpolicyadmin/client?identityDomainName=OAADomain' --header
   'Content-Type: application/json' --header 'Authorization: Basic
   <ENCODED_OAMADMIN>'
   $ curl --location --request GET 'http://
   <OAuth_Host>:<OAuth_port>/oam/services/rest/ssa/api/v1/
   oauthpolicyadmin/application?identityDomainName=OAADomain' --header
   'Content-Type: application/json' --header 'Authorization: Basic
   <ENCODED_OAMADMIN>'
   ```

3.10 Cleaning Up Installation

Perform the following steps to clean up an OAA or OAA-OARM installation completely.

1. From the installation host, connect to the management container and delete the file based vault and the logs from their respective NFS mounts:

   ```
   kubectl exec -n <namespace> -ti oaamgmt-oaa-mgmt-7d7597c694-tzzdz
   -- /bin/bash
   $ rm -rf /u01/oracle/logs/*
   $ rm -rf /u01/oracle/service/store/oaa/.*
   $ exit
   ```

2. Run the following to find the helm charts installed:

   ```
   helm ls -n <namespace>
   ```

   For example:

   ```
   helm ls -n oaans
   ```

The output will look similar to the following:

```
NAME            NAMESPACE       REVISION        UPDATED   STATUS
CHART                 APP VERSION
oaainstall      oaans           1               <date>    deployed
oaa-1.0.0-<tag>       0.1.0
oaamgmt         oaans           1               <date>    deployed
oaa-mgmt-1.0.0-<tag>  0.1.0
```

Delete the OAA charts:

```
helm delete oaainstall -n oaans
helm delete oaamgmt -n oaans
```

3. Perform the following steps to delete `coherence-operator`:

   ```
   helm delete coherence-operator -n coherence
   ```

   ```
   kubectl get sts
   ```

   ```
   kubectl get coherence.coherence.oracle.com
   ```

   ```
   kubectl delete mutatingwebhookconfigurations coherence-operator-mutating-
   webhook-configuration
   ```

4. Outside the container, run:

   ```
   kubectl get pods -n oaans
   ```

   ```
   kubectl get pods -n coherence
   ```

   If any pods remain then run:

   ```
   kubectl delete <pod_name> -n <namespace>
   ```

5. Delete the OAuth client and resources:

   a. Encode the OAM administrator user and its password by using the command:

      ```
      echo -n <username>:<password> | base64
      ```

      For example:

      ```
      echo -n weblogic:<password> | base64
      ```

      This value should be used for `<ENCODED_OAMADMIN>` in the examples below.

b. Delete the OAuth Client. For example:

```
curl --location --request DELETE 'http://
<OAuth_Host>:<OAuth_port>/oam/services/rest/ssa/api/v1/
oauthpolicyadmin/client?
name=OAAClient&identityDomainName=OAADomain' \
--header 'Authorization: Basic <ENCODED_OAMADMIN>'
```

c. Delete the OAuth Resource Server. For example:

```
curl --location --request DELETE 'http://
<OAuth_Host>:<OAuth_port>/oam/services/rest/ssa/api/v1/
oauthpolicyadmin/application?
name=OAAResource&identityDomainName=OAADomain' \
--header 'Authorization: Basic <ENCODED_OAMADMIN>'
```

d. Delete the OAuth Domain. For example:

```
curl --location --request DELETE 'http://
<OAuth_Host>:<OAuth_port>/oam/services/rest/ssa/api/v1/
oauthpolicyadmin/oauthidentitydomain?name=OAADomain' \
--header 'Authorization: Basic <ENCODED_OAMADMIN>'
```

6. Drop the database schemas as follows:

```
sqlplus sys/<password> as SYSDBA

alter session set "_oracle_script"=TRUE; ** Required for PDB’s only
**

drop user <OAA_RCU_PREFIX>_OAA cascade;
delete from SCHEMA_VERSION_REGISTRY where comp_name='Oracle
Advanced Authentication' and OWNER=UPPER('<OAA_RCU_PREFIX>_OAA');

commit;

set pages 0
set feedback off
spool /tmp/drop_directories.sql
select 'drop directory '||directory_name||';' from all_directories
where directory_name like 'EXPORT%'
/
spool off
@/tmp/drop_directories
```

7. In order to repeat the pull/tag/push of the OAA images, remove the directory `$WORKDIR/oaaimages/oaa-install/oaainstall-tmp` before rerunning the `installManagementContainer.sh` script.

4

Installing OAA and OARM Using NGINX Ingress

OAA and OARM installation supports installing and using Ingress in the following ways:

- **Installing Ingress Controller during OAA and OARM Installation**

  If you install an ingress controller as part of the OAA and OARM installation, the controller is installed on one of the nodes of the cluster and listens on a random port. For example, `https://worker1.example.com:30505`.

  The certificates generated are self signed example certificates.

  For details, see Installing Ingress Controller during OAA and OARM Installation
- **Installing your own Ingress Controller**

  If you need to use your own certificates, you can install your own Ingress controller.

  For details, see Installing your own Ingress Controller

If after installing Ingress you want to front end the Ingress with another gateway, see Appendix A: Other Considerations.

4.1 Installing Ingress Controller during OAA and OARM Installation

Update the `installOAA.properties` to install an Ingress Controller during OAA and OARM installation.

To install OAA and OARM using ingress, you must edit the `installOAA.properties` file and update the Optional Configuration section with ingress properties.

Note:

Ensure that you comment out the NodePort properties.

The following example shows the ingress properties that needs to be updated in the `installOAA.properties` file to install an ingress controller using NodePort.

```
##################################### 6. Optional
configuration#########################################
install.global.ingress.enabled=true

## All the other properties in 6.Optional configuration section must be
commented out.

##################################### 7. Ingress
```

```
configuration#########################################
#Kubernetes name space which will be used to install ingress
ingress.install=true
ingress.namespace=ingress-nginx
#Admissions controller can be installed seperately.
#Ingress admissions name is not present the the
controller.admissionWebhooks.enabled will be set to false in the nginx
ingress chart.
#ingress.admissions.name=ingress-nginx-controller-admission
#Ingress class name that would be used for installation. Must not be
exisiting
ingress.class.name=ingress-nginx-class
ingress.service.type=NodePort

#anything starting with ingress.install can be additionally supplied
to set the ingress chart value.
#ingress.install.releaseNameOverride=base
```

Note:

- `ingress.namespace` creates a namespace called `ingress-nginx`. You can change this to a name of your choice and the namespace is created for you.
- `ingress.service.type=NodePort` must be set if using a bare metal cluster. If you are using a Managed Service for your Kubernetes cluster, for example Oracle Kubernetes Engine (OKE) on Oracle Cloud Infrastructure (OCI), and connect from a browser to the Load Balancer IP address then use `ingress.service.type=LoadBalancer` property. This instructs the Managed Service to setup a Load Balancer to direct traffic to the NGINX ingress.

4.2 Installing your own Ingress Controller

Install your own Ingress controller and install OAA and OARM to use that controller.

This section provides only the additional steps or the changes required for installing OAA and OARM using NGINX ingress with nodeport.

Topics

- Prerequisites for Installing OAA using Ingress
- Updating the Install Properties File for Installing OAA/OARM Using Ingress

4.2.1 Prerequisites for Installing OAA using Ingress

If using your own Ingress controller, you must install it before installing OAA. Perform the following steps to install NGINX ingress controller on one of the nodes in the cluster.

1. Generate SSL Certificate
   a. Generate a private key (`tls.key`) and certificate signing request (CSR) using a tool of your choice. Send the CSR to your certificate authority (CA) to generate the certificate (`tls.crt`). Instructions on how to do this can be found under **Using a third party CA for generating certificates** in Generating Server Certificates and Trusted Certificates.
      Alternatively, to use a certificate for testing purposes you can generate a self-signed certificate using openssl:

      ```
      mkdir /OAA/ingress_ssl
      ```

      ```
      cd /OAA/ingress_ssl
      ```

      ```
      openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout tls.key -
      out tls.crt -subj "/CN=node.example.com"
      ```

      Note:

      If you created your own CA in Generating Server Certificates and Trusted Certificates, you can also generate a certificate using that CA.

      Note:

      The CN must match the host.domain of the kubernetes node you are installing on to prevent hostname problems during certificate verification.

   b. Create a secret for SSL by running the following command:

      ```
      kubectl create secret tls oaa-tls-cert --key /OAA/ingress_ssl/tls.key
      --cert /OAA/ingress_ssl/tls.crt
      ```

2. Install NGINX ingress
   a. Add the helm chart repository for NGINX using the following command

      ```
      helm repo add stable https://kubernetes.github.io/ingress-nginx
      ```

   b. Update the repository using the following command

      ```
      helm repo update
      ```

   c. Create a namespace, for example `nginxssl`:

      ```
      kubectl create namespace nginxssl
      ```

d. Install NGINX using the `helm install nginx-ingress` command. For example:

```
helm install nginx-ingress -n nginxssl --set
controller.extraArgs.default-ssl-certificate=oaa-tls-cert  --set
controller.service.type=NodePort --set
controller.admissionWebhooks.enabled=false stable/ingress-nginx
```

e. Get the nodeport port number. For example:

```
kubectl get services -n nginxssl -o
jsonpath="{.spec.ports[1].nodePort}" nginx-ingress-ingress-nginx-
controller
```

The command will return the port number, for example `31281`. Therefore, the URL for the ingress controller is `http://node.example.com:31281`

The hostname and port returned is used for the `install.global.serviceurl` parameter in the `installOAA.properties` in Updating the Install Properties File for Installing OAA/OARM Using Ingress

4.2.2 Updating the Install Properties File for Installing OAA/OARM Using Ingress

To install OAA and/or OARM using ingress, you must edit the `installOAA.properties` and update the `Optional Configuration` section with the ingress properties.

The following example shows the ingress properties that need to be updated in the `Optional Configuration` section of the `installOAA.properties`.

For more information, see Optional Configuration

```
##################################### 6. Optional
configuration############################################
##Ingress properties that can be used to enable ingress to the
deployment
install.global.ingress.enabled=true

#if load balancer/ingress url is present, then configure the url here.
All UI service will be behind this load balancer/ingress.
#In case ingress installation is set to true, the appropriate service
url will be fetch after ingress installation
# and will be used as service url. If provided, service url from the
property below will have higher priority.
install.global.serviceurl=https://node.example.com:31281
```

Note:

All other properties in this section, including the node port properties should be commented out.

In the `Ingress Configuration` section set `ingress.install=false`:

```
##################################### 7. Ingress
configuration##########################################
#Kubernetes name space which will be used to install ingress
ingress.install=false
```

Note: The settings above are based on a basic ingress setup. More complex scenarios are achievable using the ingress properties in Optional Configuration.

4.3 Appendix A: Other Considerations

For custom installations where the ingress is front ended by another gateway, then the following endpoints must be made available through the gateway:

```
/oaa/runtime
/oaa-policy
/policy
/oaa/rui
/oaa/authnui
/oaa-admin
/admin-ui
/oaa-email-factor
/oaa-sms-factor
/oaa-totp-factor
/oaa-push-factor
/oaa-yotp-factor
/fido
/oaa-kba
/risk-analyzer
/risk-cc
```

Each of the above endpoints must map to the ingress host and port of OAA.

Part III
Upgrading OAA and OARM

5 Upgrading OAA from Releases Earlier than 122141-20220419

Upgrade OAA from releases earlier than 122141-20220419.

To upgrade from OAA releases earlier than 122141-20220419 to a later release, follow the sections below:

- Preparing OAA and OARM Environment for Upgrade
- Performing the Upgrade

5.1 Preparing OAA and OARM Environment for Upgrade

Before starting an OAA and OARM, you must prepare your existing environment for the upgrade.

1. Make sure the following prerequisite steps are met:
   - The Kubernetes cluster must be upgraded to meet the minimum version requirements outlined in Document ID 2723908.1 on My Oracle Support.
   - The OAA and OARM deployment must be up and running.
   - Create NFS volumes as per Configuring NFS Volumes.

     Note:

     The `<NFS_VAULT_PATH>` should already exist in your existing installation so there is no need to create this.

   - Configure CoreDNS in your Kubernetes cluster as per Configuring CoreDNS for External Hostname Resolution.
   - Make sure the installation host prerequisites are met as per Installation Host Requirements
   - The installation host must have access to the NFS volumes created in Configuring NFS Volumes.
   - Download and extract the latest OAA installation files to the installation host as per Downloading Installation Files and Preparing the Management Container.

     Note:

     When you unzip the file it will create the `oaa-install` directory, for example `$WORKDIR/oaaimages/oaa-install`. This directory will be used in the steps that follow.

2. Copy the existing files from the Docker volumes to the NFS volumes:

   Note:

   If the directories for the Docker volumes are owned by `root` you have to execute the following steps as `root`.

   a. Copy the install configuration file(s) from `OAAsettings` to `<NFS_CONFIG_PATH>`:

   ```
    sudo cp -R `docker inspect OAAsettings | grep "Mountpoint" |
   cut -d':' -f2 | tr
         -d "\",\ "`/* <NFS_CONFIG_PATH>/.
   ```

   b. Copy the logs from `OAAlogs` to `<NFS_LOGS_PATH>`:

   ```
   sudo cp -R `docker inspect OAAlogs | grep "Mountpoint" | cut -
   d':' -f2 | tr -d "\",\ "`/*
         <NFS_LOGS_PATH>/.
   ```

3. Copy the `cert.p12` and `trust.p12` from `OAAcredentials` to `$WORKDIR/oaaimages/oaa-install` on the installation host:

   ```
   sudo cp `docker inspect OAAcredentials | grep "Mountpoint" | cut -
   d':' -f2 | tr -d "\",\
         "`/trust.p12 $WORKDIR/oaaimages/oaa-install/.
   ```

4. Move `installOAA.properties` from `<NFS_CONFIG_PATH>` to `$WORKDIR/oaaimages/oaa-install` on the installation host:

   ```
   sudo mv <NFS_CONFIG_PATH>/installOAA.properties $WORKDIR/oaaimages/
   oaa-install/.
   ```

   Note:

   If you have an OARM only install then run:

   ```
   sudo mv <NFS_CONFIG_PATH>/installRisk.properties $WORKDIR/oaaimages/
   oaa-install/installOAA.properties
   ```

5. Ensure all the files in the NFS volume have `rwx` access for all:

   ```
   sudo chmod -R 777 <NFS_LOGS_PATH>
   sudo chmod -R 777 <NFS_CONFIG_PATH>
   ```

6. Edit the `$WORKDIR/oaaimages/oaa-install/installOAA.properties` and change the following:

a. `install.global.image.tag` to the new `<image_tag>`. The new image tag can be found by looking at the equivalent parameter in the `$WORKDIR/oaaimages/oaa-install/installOAA.properties.template` file.

b. Add the following section to the bottom of the file and edit the values according to your NFS server (hostname or IP address) and volumes:

```
#################################### 8. OAA management
configuration############################################
#NFS volumes
install.mount.config.path=<NFS_CONFIG_PATH>
install.mount.config.server=<NFS_CONFIG_SERVER>
install.mount.creds.path=<NFS_CREDS_PATH>
install.mount.creds.server=<NFS_CREDS_SERVER>
install.mount.logs.path=<NFS_LOGS_PATH>
install.mount.logs.server=<NFS_LOGS_SERVER>
install.mgmt.release.name=oaamgmt
common.local.sslcert=<LOCAL_PATH>/cert.p12
common.local.trustcert=<LOCAL_PATH>/trust.p12
```

Note:

For more details on these parameters, see Management Container Configuration.

Next Steps: Performing the Upgrade.

5.2 Performing the Upgrade

Upgrade the existing OAM and OARM installation to the latest version.

1. On the installation host where you downloaded the installation files, navigate to the `$WORKDIR/oaaimages/oaa-install` directory:

```
cd $WORKDIR/oaaimages/oaa-install
```

2. Run the `installManagementContainer.sh` script as follows:

```
./installManagementContainer.sh -t ./<oaa-image>.tar
```

As the upgrade progresses you will be prompted to answer various questions and perform certain tasks. The table below outlines some of the questions or tasks you may be asked to answer or perform:

Output	Action
`Version check failed for podman, would you like to use docker instead? [Y/N]:`	This message will appear if the installation host does not have podman installed as per the Installation Host Requirements. If you do not have podman at the required version then choose to use Docker by answering "`Y`".

Output	Action

`Login into container-registry.oracle.com from browser and navigate to the location of each of the supporting images. On the right-hand side, select the Language from the drop-down menu and click Continue. Read the Oracle Standard Terms and Restrictions and click Accept to agree. Finally, use 'podman login' to store the credentials locally. Login successful? [Y/N]:`
 	
 Note:
 If using Docker the above will say `docker login`.
	• Launch a browser and visit https://container-registry.oracle.com. Sign in with your credentials. Navigate to **Database** and then **instantclient**. On the right-hand side, select the Language from the drop-down menu and click **Continue**. Read the Oracle Standard Terms and Restrictions and click **Accept** to agree.
	 • Depending on whether you are using podman or Docker use `podman login` or `docker login` to log into `container-registry.oracle.com` and enter your credentials when prompted:
	 `podman login container-registry.oracle.com`
	 or:
	 `docker login container-registry.oracle.com`

`Use 'podman login' to login into your private registry if you have not done so previously.Login successful? [Y/N]:`
 	
 Note:
 If using Docker the above will show `docker login`
	If your private Container Image Registry (CIR) where you store images requires a login, use `podman login` or `docker login` to log into the CIR and enter your credentials when prompted:
	 `podman login <container-registry.example.com>`
	 or:
	 `docker login <container-registry.example.com>`

`Would you like to specify a kube context (otherwise 'kubernetes-admin@kubernetes' will be used)? [Y/N]:`	If you have multiple kube contexts in your cluster you can choose which context to use. If you select "`Y`" you must type the context you wish to use. If you wish to use the default context chosen, or only have one context in your kube config choose "`N`".

Note:

The table above does not include an exhaustive list of all prompts you will see during the upgrade as most are self explanatory.

3. Once the Management Container installation is complete you will see output similar to the following:

   ```
   Release "oaamgmt" has been upgraded. Happy Helming!
   NAME: oaamgmt
   LAST DEPLOYED: <date>
   NAMESPACE: oaans
   STATUS: deployed
   REVISION: 2
   TEST SUITE: None
   Waiting 15 secs for OAA mgmt deployment to start...
   File /u01/oracle/scripts/settings/installOAA.properties already exists,
   copying file to /u01/oracle/scripts/settings/installOAA.properties.<date>
   Copying OAA properties file to oaans/oaamgmt-oaa-mgmt-7d7597c694-
   tzzdz:/u01/oracle/scripts/settings
   Use command 'kubectl exec -n oaans -ti oaamgmt-oaa-mgmt-7d7597c694-tzzdz
   -- /bin/bash' to get a shell to the OAA mgmt pod.
   From pod shell, use command 'kubectl get pods' to verify communication
   with the cluster.
   Continue OAA installation from the OAA mgmt pod.
   OAA management installation complete.
   ```

4. As per the output connect to the OAA management pod, for example:

   ```
   kubectl exec -n oaans -ti oaamgmt-oaa-mgmt-7d7597c694-tzzdz -- /bin/bash
   ```

   This will take you into a bash shell inside the OAA management pod:

   ```
   [oracle@oaamgmt-oaa-mgmt-7d7597c694-tzzdz /]$
   ```

5. Inside the OAA management pod perform the upgrade:

   ```
   [oracle@oaamgmt-oaa-mgmt-7d7597c694-tzzdz /]$ cd ~
   [oracle@oaamgmt-oaa-mgmt-7d7597c694-tzzdz ~]$ ./OAA.sh -f
   installOAA.properties
   ```

   Note:

   This will use the `<NFS_CONFIG_PATH>/installOAA.properties` file.

6. When the upgrade gets as far as the helm sanity checks, you will see all the helm checks pass except for `'<common.deployment.name>-sanity-check`. For example:

   ```
   Release "oaainstall" has been upgraded. Happy Helming!
   NAME: oaainstall
   LAST DEPLOYED: Thu Apr 14 20:43:14 2022
   NAMESPACE: dev
   STATUS: deployed
   REVISION: 2
   NOTES:
   Get the Oracle Advance Authentication(OAA) Service URL by running these
   commands:
   ```

```
Waiting for OAA installation check : oaainstall
NAME: oaainstall
Error: pod oaainstall-sanity-check failed
LAST DEPLOYED: Thu Apr 14 20:43:14 2022
NAMESPACE: dev
STATUS: deployed
REVISION: 2
TEST SUITE:     oaainstall-sanity-check
Last Started:   Thu Apr 14 20:45:01 2022
Last Completed: Thu Apr 14 20:46:54 2022
Phase:          Failed
TEST SUITE:     oaainstall-email-sanity-check
Last Started:   Thu Apr 14 20:44:14 2022
Last Completed: Thu Apr 14 20:44:16 2022
Phase:          Succeeded
TEST SUITE:     oaainstall-fido-sanity-check
Last Started:   Thu Apr 14 20:44:16 2022
Last Completed: Thu Apr 14 20:44:18 2022
Phase:          Succeeded
TEST SUITE:     oaainstall-oaa-admin-ui-sanity-check
Last Started:   Thu Apr 14 20:44:18 2022
Last Completed: Thu Apr 14 20:44:20 2022
Phase:          Succeeded
TEST SUITE:     oaainstall-uaskbadb-test-connection
Last Started:   Thu Apr 14 20:43:23 2022
Last Completed: Thu Apr 14 20:43:43 2022
Phase:          Succeeded
TEST SUITE:     oaainstall-oaa-kba-sanity-check
Last Started:   Thu Apr 14 20:44:20 2022
Last Completed: Thu Apr 14 20:44:22 2022
Phase:          Succeeded
TEST SUITE:     oaainstall-uaspoledb-test-connection
Last Started:   Thu Apr 14 20:43:43 2022
Last Completed: Thu Apr 14 20:44:14 2022
Phase:          Succeeded
TEST SUITE:     oaainstall-oaa-policy-sanity-check
Last Started:   Thu Apr 14 20:44:22 2022
Last Completed: Thu Apr 14 20:44:24 2022
Phase:          Succeeded
TEST SUITE:     oaainstall-push-sanity-check
Last Started:   Thu Apr 14 20:44:24 2022
Last Completed: Thu Apr 14 20:44:26 2022
Phase:          Succeeded
TEST SUITE:     oaainstall-riskdb-test-connection
Last Started:   Thu Apr 14 20:43:18 2022
Last Completed: Thu Apr 14 20:43:20 2022
Phase:          Succeeded
TEST SUITE:     oaainstall-risk-cc-sanity-check
Last Started:   Thu Apr 14 20:44:26 2022
Last Completed: Thu Apr 14 20:44:28 2022
Phase:          Succeeded
TEST SUITE:     oaainstall-risk-sanity-check
Last Started:   Thu Apr 14 20:44:28 2022
Last Completed: Thu Apr 14 20:44:30 2022
Phase:          Succeeded
```

```
TEST SUITE:     oaainstall-uasdb-test-connection
Last Started:   Thu Apr 14 20:43:20 2022
Last Completed: Thu Apr 14 20:43:23 2022
Phase:          Succeeded
TEST SUITE:     oaainstall-sms-sanity-check
Last Started:   Thu Apr 14 20:44:30 2022
Last Completed: Thu Apr 14 20:44:32 2022
Phase:          Succeeded
TEST SUITE:     oaainstall-spui-sanity-check
Last Started:   Thu Apr 14 20:44:32 2022
Last Completed: Thu Apr 14 20:44:34 2022
Phase:          Succeeded
TEST SUITE:     oaainstall-totp-sanity-check
Last Started:   Thu Apr 14 20:44:34 2022
Last Completed: Thu Apr 14 20:44:59 2022
Phase:          Succeeded
TEST SUITE:     oaainstall-yotp-sanity-check
Last Started:   Thu Apr 14 20:44:59 2022
Last Completed: Thu Apr 14 20:45:01 2022
Phase:          Succeeded
NOTES:
Get the Oracle Advance Authentication(OAA) Service URL by running these
commands:
Helm test failed
```

At this point you must perform the following to clean up `sts`:

a. Run the following command to delete `sts`:

```
kubectl delete sts -l coherenceCluster=<common.deployment.name> -n
<common.kube.namespace>
```

Note:

The value of `<common.deployment.name>` and `<common.kube.namespace>` can be found in the `NFS_CONFIG_PATH/installOAA.properties`.

b. Run the following command to check the status of `sts`:

```
kubectl get sts -n <common.kube.namespace>
```

Once the status of `sts` shows `Ready(n/n)` you can continue to the next step:

```
NAME                   READY   AGE
oaainstall-cache-rest  1/1     5m
oaainstall-storage     3/3     5m
```

c. Rerun `./OAA.sh -f installOAA.properties` and this time the helm tests should all pass.

7. Once the upgrade is complete run the following commands to restart the pods:

```
kubectl get deploy -n <common.kube.namespace> -l app.kubernetes.io/
instance=<common.deployment.name> -o name | xargs -n 1 kubectl
rollout restart -n <common.kube.namespace>
```

8. Run the following command to make sure all the pods are running:

```
kubectl get pods -n <namespace>
```

For example:

```
kubectl get pods -n oaans
```

The output should look similar to the following:

```
NAME                                       READY   STATUS
RESTARTS   AGE
oaainstall-cache-rest-0                    1/1     Running
0          2m30s
oaainstall-email-5465c65844-j2pqc          1/1     Running
0          2m36s
oaainstall-fido-7f4c455559-g6ppz           1/1     Running
0          2m36s
oaainstall-oaa-7b5574f9bd-wt5rf            1/1     Running
0          2m35s
oaainstall-oaa-admin-ui-64b587dc4c-ftggm   1/1     Running
0          2m36s
oaainstall-oaa-kba-6859456966-z6rbc        1/1     Running
0          2m36s
oaainstall-oaa-policy-6978575cf-hrw6b      1/1     Running
0          2m36s
oaainstall-push-7ff9c4d76b-hz7pl           1/1     Running
0          2m36s
oaainstall-risk-6b55687d7-jpvcs            1/1     Running
0          2m36s
oaainstall-risk-cc-99b545467-tgw62         1/1     Running
0          2m36s
oaainstall-sms-589fc9d4b5-bfqp9            1/1     Running
0          2m36s
oaainstall-spui-b9bfccfc9-f7rkt            1/1     Running
0          2m35s
oaainstall-storage-0                       1/1     Running
0          23m
oaainstall-storage-1                       1/1     Running
0          23m
oaainstall-storage-2                       1/1     Running
0          23m
oaainstall-totp-58987cc8d7-hmh29           1/1     Running
0          2m35s
oaainstall-yotp-797754fcfb-f294x           1/1     Running
0          2m35s
```

```
oaamgmt-oaa-mgmt-7d7597c694-tzzdz          1/1     Running   0
3m42s
```

Note:

It will take a few minutes for all the pods to show a `READY` status of `1/1`.

6 Upgrading OAA from 122141-20220419 onwards

Upgrade OAA from 122141-20220419.

To upgrade from OAA 122141-20220419 onwards, to the latest release follow the sections below:

- Preparing OAA and OARM Environment for Upgrade
- Performing the Upgrade

6.1 Preparing OAA and OARM Environment for Upgrade

Before starting an OAA and OARM, you must prepare your existing environment for the upgrade.

1. The Kubernetes cluster must be upgraded to meet the minimum version requirements outlined in Document ID 2723908.1 on My Oracle Support.
2. The OAA and OARM deployment must be up and running.
3. Make sure the installation host prerequisites are met as per Installation Host Requirements.
4. The installation host must have access to the NFS volumes for the existing deployment.See Configuring NFS Volumes.
5. Download and extract the latest OAA installation files to the installation host as per Downloading Installation Files and Preparing the Management Container.

   Note:

   When you unzip the file it will create the `oaa-install` directory, for example `$WORKDIR/oaaimages/oaa-install`. This directory will be used in the steps that follow.

6. Find the name of the existing OAA Management container:

   ```
   kubectl get pods -n <namespace> | grep oaamgmt
   ```

   For example:

   ```
   kubectl get pods -n oaans | grep oaamgmt
   ```

   The output will look similar to the following:

   ```
   oaamgmt-oaa-mgmt-bf6d5c88-29lrn           1/1     Running    0           64d
   ```

7. Copy the `installOAA.properties` for the current deployment to the `$WORKDIR/oaaimages/oaa-install` directory:

   ```
   kubectl cp <namespace>/<oaamgmt-pod>:/u01/oracle/scripts/settings/
   installOAA.properties $WORKDIR/oaaimages/oaa-install/
   installOAA.properties
   ```

   For example:

   ```
   kubectl cp oaans/oaamgmt-oaa-mgmt-bf6d5c88-29lrn:/u01/oracle/
   scripts/settings/installOAA.properties $WORKDIR/oaaimages/oaa-
   install/installOAA.properties
   ```

8. Edit the `$WORKDIR/oaaimages/oaa-install/installOAA.properties` and change the `install.global.image.tag` to the image tag for the latest release.

   > **Note:**
   >
   > You can find the correct value for `install.global.image.tag` in the `$WORKDIR/oaaimages/oaa-install/installOAA.properties.template`.

9. Ensure that the OAA Management helm chart (`oaamgmt`) is visible by running the following command:

   ```
   helm ls -n <namespace> | grep oaamgmt
   ```

   For example:

   ```
   helm ls -n oaans | grep oaamgmt
   ```

   The output should look similar to the following:

   ```
   oaamgmt   oaans   1   <DATE> <TIME> +0000 UTC deployed   oaa-
   mgmt-1.0.0-12.2.1.4.1-<TAG>   0.1.0
   ```

Next Steps: Performing the Upgrade.

6.2 Performing the Upgrade

Upgrade the existing OAM and OARM installation to the latest version.

1. On the installation host where you downloaded the installation files, navigate to the `$WORKDIR/oaaimages/oaa-install` directory:

   ```
   cd $WORKDIR/oaaimages/oaa-install
   ```

2. Run the `installManagementContainer.sh` script as follows:

   ```
   ./installManagementContainer.sh -t ./<oaa-image>.tar
   ```

   This will upgrade the installed oaamgmt chart. As the upgrade progresses you will be prompted to answer various questions and perform certain tasks. The table below outlines some of the questions or tasks you may be asked to answer or perform:

Output	Action
`Version check failed for podman, would you like to use docker instead? [Y/N]:`	This message will appear if the installation host does not have podman installed as per the Installation Host Requirements. If you do not have podman at the required version then choose to use Docker by answering "`Y`".

`Login into container-registry.oracle.com from browser and navigate to the location of each of the supporting images. On the right-hand side, select the Language from the drop-down menu and click Continue. Read the Oracle Standard Terms and Restrictions and click Accept to agree. Finally, use 'podman login' to store the credentials locally. Login successful? [Y/N]:`
 	
 Note: If using Docker the above will say `docker login`.
	• Launch a browser and visit https://container-registry.oracle.com. Sign in with your credentials. Navigate to **Database** and then **instantclient**. On the right-hand side, select the Language from the drop-down menu and click **Continue**. Read the Oracle Standard Terms and Restrictions and click **Accept** to agree.
	• Depending on whether you are using podman or Docker use `podman login` or `docker login` to log into `container-registry.oracle.com` and enter your credentials when prompted:
	
	 `podman login container-registry.oracle.com`
	
	 or:
	
	 `docker login container-registry.oracle.com`

`Use 'podman login' to login into your private registry if you have not done so previously.Login successful? [Y/N]:`
 	
 Note: If using Docker the above will show `docker login`
	If your private Container Image Registry (CIR) where you store images requires a login, use `podman login` or `docker login` to log into the CIR and enter your credentials when prompted:
	
	 `podman login <container-registry.example.com>`
	
	 or:
	
	 `docker login <container-registry.example.com>`

Note:

The table above does not include an exhaustive list of all prompts you will see during the upgrade as most are self explanatory.

3. Once the Management Container installation is complete you will see output similar to the following:

```
Release "oaamgmt" has been upgraded. Happy Helming!
NAME: oaamgmt
LAST DEPLOYED: <DATE>
NAMESPACE: oaans
STATUS: deployed
REVISION: 2
TEST SUITE: None
Waiting 15 secs for OAA mgmt deployment to start...
Executing 'kubectl get pods  -n oaans | grep oaamgmt-oaa-mgmt- '...
oaamgmt-oaa-mgmt-6f4c9cd56f-std6l         0/1
ContainerCreating   0          15s
oaamgmt-oaa-mgmt-bf6d5c88-29lrn           1/1
Running             0          4h42m
Waiting 15 secs for OAA mgmt rollout to complete...
Executing 'kubectl get pods  -n oaans | grep oaamgmt-oaa-mgmt- '...
oaamgmt-oaa-mgmt-6f4c9cd56f-std6l         0/1
ContainerCreating   0          31s
oaamgmt-oaa-mgmt-bf6d5c88-29lrn           1/1
Running             0          4h42m
Waiting 15 secs for OAA mgmt rollout to complete...
Executing 'kubectl get pods  -n oaans | grep oaamgmt-oaa-mgmt- '...
oaamgmt-oaa-mgmt-6f4c9cd56f-std6l         1/1     Running
0          46s
oaamgmt-oaa-mgmt-bf6d5c88-29lrn           1/1     Terminating
0          4h42m
Waiting 15 secs for OAA mgmt rollout to complete...
Executing 'kubectl get pods  -n oaans | grep oaamgmt-oaa-mgmt- '...
oaamgmt-oaa-mgmt-6f4c9cd56f-std6l         1/1     Running
0          61s
oaamgmt-oaa-mgmt-bf6d5c88-29lrn           1/1     Terminating
0          4h43m
Waiting 15 secs for OAA mgmt rollout to complete...
Executing 'kubectl get pods  -n oaans | grep oaamgmt-oaa-mgmt- '...
oaamgmt-oaa-mgmt-6f4c9cd56f-std6l         1/1     Running
0          76s
oaamgmt-oaa-mgmt-bf6d5c88-29lrn           1/1     Terminating
0          4h43m
Waiting 15 secs for OAA mgmt rollout to complete...
File /u01/oracle/scripts/settings/installOAA.properties already
exists, copying file to /u01/oracle/scripts/settings/
installOAA.properties.<DATE>
Copying OAA properties file to oaans/oaamgmt-oaa-mgmt-6f4c9cd56f-
std6l:/u01/oracle/scripts/settings
Use command 'kubectl exec -n oaans -ti oaamgmt-oaa-mgmt-6f4c9cd56f-
std6l -- /bin/bash' to get a shell to the OAA mgmt pod.
From pod shell, use command 'kubectl get pods' to verify
```

```
communication with the cluster.
Continue OAA installation from the OAA mgmt pod.
OAA management installation complete.
```

4. As per the output connect to the new OAA management pod, for example:

   ```
   kubectl exec -n oaans -ti oaamgmt-oaa-mgmt-6f4c9cd56f-std6l -- /bin/bash
   ```

   This will take you into a bash shell inside the OAA management pod:

   ```
   [oracle@oaamgmt-oaa-mgmt-6f4c9cd56f-std6l /]$
   ```

5. Inside the OAA management pod perform the upgrade:

   ```
   [oracle@oaamgmt-oaa-mgmt-6f4c9cd56f-std6l /]$ cd ~
   [oracle@oaamgmt-oaa-mgmt-6f4c9cd56f-std6l ~]$ ./OAA.sh -f
   installOAA.properties
   ```

   Note:

   This will use the `<NFS_CONFIG_PATH>/installOAA.properties` file.

6. Once the upgrade is complete you should see the upgrade is successful and the deployment details are printed to the screen.
7. Run the following command to make sure all the pods are running:

   ```
   kubectl get pods -n <namespace>
   ```

   For example:

   ```
   kubectl get pods -n oaans
   ```

   The output should look similar to the following:

   ```
   NAME                                                        READY   STATUS
   RESTARTS   AGE
   oaainstall-customfactor-b5cf55778-rwg5l                     1/1     Running
   0          6m29s
   oaainstall-email-65dc5f679-6xtmd                            1/1     Running
   0          6m29s
   oaainstall-fido-5b46884c68-q9dxp                            1/1     Running
   0          6m29s
   oaainstall-oaa-65779f845b-b9g6c                             1/1     Running
   0          6m29s
   oaainstall-oaa-admin-ui-6689c9d4cd-jhfzx                    1/1     Running
   0          6m29s
   oaainstall-oaa-kba-6b8c4cfb-x2xsm                           1/1     Running
   0          6m28s
   oaainstall-oaa-policy-7997547c98-8jl4n                      1/1     Running
   ```

```
0           6m28s
oaainstall-push-58b478c4f9-fx95n                      1/1
Running    0           6m28s
oaainstall-risk-68bf8b75b7-gg99q                      1/1
Running    0           6m28s
oaainstall-risk-cc-6f669d5c5c-sfhfx                   1/1
Running    0           6m28s
oaainstall-sms-786d684994-lktfz                       1/1
Running    0           6m28s
oaainstall-spui-94f6f5f9b-pmhc2                       1/1
Running    0           6m28s
oaainstall-totp-7759f4598d-8rqwv                      1/1
Running    0           6m27s
oaainstall-yotp-5f865df96-c5x52                       1/1
Running    0           6m27s
oaamgmt-oaa-mgmt-859d65684f-jq84f                     1/1
Running    0           48m
oaainstall-riskdb-6f57c69b-dtmrd                      1/1
Running    0           50m
```

7

Rolling Back the Upgrade

If the upgrade fails, you can rollback the failed upgrade, fix the issue, and then retry the upgrade. You can also rollback if the upgrade was successful but you subsequently have functional issues.

1. Check the history of the OAA chart using the following command:

   ```
    helm history oaainstall -n <namespace>
   ```

   For example:

   ```
    helm history oaainstall -n oaans
   ```

   The output will look similar to the following:

   ```
   $ helm history oaainstall -n oaans
   REVISION        UPDATED      STATUS
   CHART                           APP VERSION     DESCRIPTION
   1               <date>       superseded
   oaa-1.0.0-12.2.1.4.1-20220127   0.1.0           Install complete
   2               <date>       superseded      oaa-1.0.0-
   latest                0.1.0           Upgrade complete
   ```

2. Rollback to the OAA chart release to the previous version using the command:

   ```
   helm rollback oaainstall -n <namespace>
   ```

   For example:

   ```
   helm rollback oaainstall -n oaans
   ```

3. Check the history of the OAA charts using "helm history" to show the rollback has occurred:

   ```
   $ helm history oaainstall -n oaans
   REVISION        UPDATED      STATUS
   CHART                           APP VERSION     DESCRIPTION
   1               <date>       superseded
   oaa-1.0.0-12.2.1.4.1-20220127   0.1.0           Install complete
   2               <date>       superseded      oaa-1.0.0-
   latest                0.1.0           Upgrade complete
   3               <date>       deployed
   oaa-1.0.0-12.2.1.4.1-20220127   0.1.0           Rollback to 1
   ```

4. Verify the rollback is successful and all the pods are running:

   ```
   kubectl get pods -n <namespace>
   ```

   For example:

   ```
   kubectl get pods -n oaans
   ```

5. Run the following command to delete `sts`:

   ```
   kubectl delete sts -l coherenceCluster=<common.deployment.name> -n
   <common.kube.namespace>
   ```

   > **Note:**
   >
   > The value of `<common.deployment.name>` and `<common.kube.namespace>` can be found in the `NFS_CONFIG_PATH/installOAA.properties`.

6. Run the following command to check the status of `sts`:

   ```
   kubectl get sts -n <common.kube.namespace>
   ```

   Once the status of `sts` shows `Ready(n/n)` you can continue to the next step:

   ```
   NAME                     READY   AGE
   oaainstall-cache-rest    1/1     5m
   oaainstall-storage       3/3     5m
   ```

7. Rerun `./OAA.sh -f installOAA.properties`.

8. Once the helm test is complete run the following commands to restart the pods:

   ```
   kubectl get deploy -n <common.kube.namespace> -l app.kubernetes.io/
   instance=<common.deployment.name> -o name | xargs -n 1 kubectl
   rollout restart -n <common.kube.namespace>
   ```

9. Run the following command to make sure all the pods are running:

   ```
   kubectl get pods -n <namespace>
   ```

   For example:

   ```
   kubectl get pods -n oaans
   ```

   > **Note:**
   >
   > It will take a few minutes for all the pods to show a `READY` status of `1/1`.

Part IV

Transitioning from Oracle Adaptive Access Manager (OAAM) to Oracle Adaptive Risk Management (OARM) and Oracle Advanced Authentication (OAA)

OAAM runtime is composed of two functional components:

- Risk Evaluation Engine
- User interaction with built-in support for data capture, and login support

With the introduction of microservices, Oracle Adaptive Risk Management (OARM) fills the role of the Risk Evaluation Engine, while Oracle Advanced Authentication (OAA) provides the support for data capture and login.

Therefore, the evolution for the existing users of OAAM is a modern service composed of OARM and OAA that leverages the existing data as-is.

OARM is designed to work off your existing data from OAAM. This means that you can wire OAA and OARM to your existing database. However, it is strongly recommended that you clone your existing database and wire OAA and OARM to the cloned DB.

OARM policy model is simplified and easier to use and it is recommended to switch to the new policy model. However, the existing policies continue to work after transitioning to OARM.

8

OAAM Features Not Supported or Changed in OARM and OAA

The following table lists the features from OAAM that are not supported in OARM and OAA:

Table 8-1 Features from OAAM that are not Supported in OARM and OAA

Features	Resolution	Additional Note
Challenge Policies	Configure the challenge methods in OAA. All the data is preserved for existing challenge methods. End users do not have to re-register.	OAA offers modern challenge methods like Passwordless authentication, FIDO2, Yubikey, and so on.
Authentication Pads	Use modern strong authentication capabilities that are available in OAA.	Authentication pads are used to mitigate capturing passwords, which can be achieved using Multi-Factor Authentication (MFA) and passwordless authentications in OAA.
OAAM Offline	None	OAAM offline is used to evaluate the effectiveness of policies. This can be better accomplished by creating policies online targeted at test users or no actions.
Customer Service Representative (CSR) / Case Management	Use REST APIs to integrate with external Customer Care Services.	Modern customer care CSR services provide vastly superior capabilities.
Investigation Management	None	None
SOAP API	Use modern REST APIs in place of SOAP APIs	Eliminates the need for client-side SDKs.
Support for Registration Flows	Switch to modern User self-service-based registration	OAA supports a modern User self-service-based registration through User Preferences Console.
Linked Entities	None	None

Additional Changes in OARM and OAA

The following are some additional changes that you find after transitioning to OAA and OARM.

- Browsing legacy OAAM policy is supported through **OAAM Policy Explorer** in the OAA Administration UI Console.
- Built in support for Device Identification policies.
- Out-of-the-box policies no longer include non-business critical policies (Authentication, mobile and social, pre-authentication, create transaction, update transaction, and so on).

- Business Transactions use cases can be accomplished using **Custom User Activity** flows.
- OAM integration use cases continue to work with OAA and OARM replacing OAAM.
- Policies associated with Post Auth Policy checkpoint in OAAM will be associated with **User Authentication** Activity.
- End-user's email, SMS and KBA questions are picked up by OAA. No changes are required.
- Factor registration is supported by OAA.
- Policies/CheckPoints: No migration required for any custom policies that administrators have authored in OAAM.

9 Procedure for Transitioning from OAAM to OAA and OARM

This section provides the procedure for transitioning from OAAM to OAA and OARM.

Topics

Transitioning from OAAM to OAA and OARM involves the following steps:

1. Preparing OAAM Environment for Transition
2. Prerequisite Configurations for Installing OAA and OARM

> **Note:**
>
> The OAA and OARM installation does not require a new database as it uses the database in the cloned OAAM installation.

3. Downloading Installation Files and Preparing the Management Container
4. Performing the OAAM to OAA and OARM Transition

9.1 Preparing OAAM Environment for Transition

To transition an Oracle Adaptive Access Management (OAAM) environment to OAA and OARM, the OAAM environment must meet the following criteria:

- OAAM must be at 11gR2PS3 with the latest Bundle Patch applied.
- If OAAM 11gR2PS3 is integrated with Oracle Access Management (OAM) and/or Oracle Identity Manager (OIM), then OAM/OIM must be 12cPS4. This involves first upgrading to 12cPS3 (12.2.1.3.0), and then upgrading to 12cPS4 (12.2.1.4.0).
    1. For details on performing an OAM 11gR2PS3 upgrade to 12cPS3, refer to Introduction to Upgrading Oracle Access Manager to 12c (12.2.1.3.0)

       For details on performing an OIG 11gR2PS3 upgrade to 12cPS3, refer to Introduction to Upgrading Oracle Identity Manager to 12c (12.2.1.3.0)

       Details on performing an integrated OIG/OAM 11gR2PS3 upgrade to 12cPS3 can also be found in the above documentation.
    2. Once the environment is upgraded to 12cPS3 then the environment must be upgraded to 12cPS4.
       For details on performing an OAM 11gR2PS3 upgrade to 12cPS4, refer to Introduction to Upgrading Oracle Access Manager to 12c (12.2.1.4.0)

       For details on performing an OIG 11gR2PS3 upgrade to 12cPS4, refer to Introduction to Upgrading Oracle Identity Manager to 12c (12.2.1.4.0)

       Details on performing an integrated OIG/OAM 11gR2PS3 upgrade to 12cPS4 can also be found in the above documentation.

- The database used for OAAM must be at a version supported by OAA: 12c (12.2.0.1+), 18c, or 19c. If the OAAM database is not at this version then the database must be upgraded before proceeding with the OAA and OARM installation.

  > **Note:**
  >
  > If upgrading the database from 12.1 to a supported version, after upgrading the database you must set the `compatible` database parameter to the version you have upgraded to. See your database version specific documentation to determine the required `compatible` value. Failure to do this will cause the OAA transition to fail.

- In order to prevent any issues and mitigate any risk to the production OAAM environment, it is recommended that you either clone the entire OAAM environment, or clone the OAAM database prior to transition to OAA and OARM. The OAA and OARM installation will connect to the OAAM schema in the cloned database, which allows the OAAM environment to operate as normal while the transition occurs, and mitigates any risk to your production OAAM environment. For details on creating a cloned environment, see Cloning Oracle Access Manager Environment

9.2 Performing the OAAM to OAA and OARM Transition

Prerequisites: Ensure you have followed the prerequisites before starting the transition. For details, see Prerequisite Configurations for Installing OAA and OARM.

Perform the following steps to transition from OAAM to OAA and OARM

1. Obtain the `bharosa.uio.default.user.group` property value from the OAAM Administration console.
   a. Login to the OAAM Administration console. For example: `http://oaam.example.com:14200/oaam_admin`.
   b. In the left hand navigation menu select **Properties** and search for the property `bharosa.uio.default.user.group`.
   c. In the **Search Results** make note of the value returned. This value will be set later for `oauth.applicationid` in `installOAA.properties`.
2. Obtain the OAAM schema details. You must have the following information prior to performing the transition
   - The hostname and listener port of the cloned OAAM database
   - The name of the OAAM schema (for example, DEV_OAAM) and the schema password
   - The SYS schema password
3. Export OAAM Config Keys from Oracle Fusion Middleware Enterprise Manager 11g.
   a. Login to the Oracle Fusion Middleware Enterprise Manager 11g for OAAM. For example, `http://oaam.example.com:7001/em`

   b. In the left hand navigation menu expand **WebLogic Domain**. Right click on the domain and select **Security** and then **Credentials**.
   c. In the Credentials pane expand **oaam** and make sure the keys `DESede_db_key_alias` and `DESede_config_key_alias` exist.
   d. Select `DESede_db_key_alias` key and click **Edit**. Make note of the value under "**Credential**."
   e. Select `DESede_config_key_alias` key and click **Edit**. Make note of the value under "**Credential**".

4. Set the following properties in the `installOAA.properties`. For details about the instalOAA.properties file, see Preparing the Properties file for OAA and OARM Installation
   a. Set `oauth.applicationid` to the value returned earlier for `bharosa.uio.default.user.group`.
   b. The following database parameters must be set to the cloned OAAM database and schemas:

```
database.createschema=false
database.host=<OAAM_DB_HOST>
database.port=<OAAM_DB_PORT>
database.sysuser=sys
database.syspassword=<SYS_PASSWORD>
database.schema=<OAAM_SCHEMA>
database.schemapassword=<OAM_SCHEMA_PASSWORD>
database.svc=<OAAM_DB_SERVICE_NAME>
database.name=<OAAM_DB_NAME>
```

   For example,

```
database.createschema=false
database.host=oaamdb.example.com
database.port=1521
database.sysuser=sys
database.syspassword=<password>
database.schema=DEV_OAAM
database.schemapassword=<password>
database.svc=oaamdb.example.com
database.name=oaamdb
```

   Note:

   `database.tablespace=DEV_OAA_TBS` is not required because `database.createschema=false`.

   c. Set the deployment mode based on the install type. Possible values are `OAA` or `Both`. Default mode is `Both`, which installs OAA integrated with Risk.
   For example:

```
common.deployment.mode=Both
```

d. Set the OAAM configuration keys:

   - Base64 encoded config key from the migrating system:`common.migration.configkey=`
     If enabled, the value is placed in the vault and used for migration of legacy data.

     Set the parameter `common.migration.configkey` to the value returned for `DESede_config_key_alias` in Oracle Fusion Middleware Enterprise Manager 11. For example:

     ```
     common.migration.configkey=Z147tibEm2iDoV5o5kwV0BUIvCo0Auxu
     ```

   - Base64 encoded db key from the migrating system: `common.migration.dbkey=`
     If enabled, the value is placed in the vault and used for migration of DB data.

     Set the parameter `common.migration.dbkey` to the value returned for `DESede_db_key_alias` in Oracle Fusion Middleware Enterprise Manager 11. For example:

     ```
     common.migration.dbkey=8b/3zUb0Bz3qIz5uwg0jUW77W3oZtVtK
     ```

e. If the OAAM environment is integrated with OIM 12cPS4 then set the following parameter:

   ```
   common.oim.integration=true
   ```

   This also enables the forgot password functionality.

f. Set the import snapshot property to `false`:

   ```
   common.deployment.import.snapshot=false
   ```

   Note:

   This is a very important step. **Don't** set this value to `true`. If you set this value to true your OAAM data will be overwrriten!

5. Deploy OAA and/or OARM. For details, see Deploying OAA and OARM

6. Set the `vcryptuser.groupid.lowercase` configuration property so that OAA and OAAM use the same groupid convention. Use the `<PolicyUrl>/policy/config/property/v1` REST API as shown in the following sample request.

   ```
   curl --location -g --request PUT '<PolicyUrl>/policy/config/
   property/v1' \
   --header 'Content-Type: application/json' \
   --header 'Authorization: Basic
   <Base64Encoded(<username>:<password>)>' \
   --data '[
   {
   "name": "vcryptuser.groupid.lowercase",
   ```

```
 "value": "false"
 }
]'
```

> **Note:**
>
> In this case remove `/oaa-policy` from the `<PolicyUrl>`, for example use `https://<host>:<port>/policy/config/property/v1` not `https://<host>:<port>/oaa-policy/policy/config/property/v1`

For details about finding the `PolicyUrl` and authenticating, see OAA Admin API.

For details about the REST API, see Configuration Properties REST Endpoints

7. If you have performed an NodePort only installation, perform post-installation tasks. See Post-Installation Steps for NodePort.
8. Follow the Post Installation Steps for OAA-OARM and OARM installs.

> **Note:**
>
> Do **NOT** follow Post Installation Steps for all installations as you will overwrite the risk policies.

9. If you were previously using an OAM-OAAM integrated environment then OAM 12cPS4 must be rewired to use OAA. For details see, Integrate Oracle Access Management with Oracle Advanced Authentication.

> **Note:**
>
> In the section `Update the WebGate to use the OAA MFA Scheme for the protected application`, update your protected applications to use the new Authentication Policy: `OAA_MFA-Policy`.

10

Viewing the Existing OAAM Policies in OAA and OARM Environment

After transitioning to the OAA and OARM environment, you can view all your existing OAAM policies on the **OAAM Policy Explorer**

OAAM Policy Explorer is integrated within the Admin UI Console. For more details about accessing the Admin Console, see Printing Deployment Details

To view your OAAM policies in the OAAM Policy Explorer:

1. Login to the OAA Administration console `https://<AdminUrl>`. You are redirected to the OAM login page, as the console is protected by OAM OAuth. Specify your credentials and login.

   Note:

   For details on finding the `<AdminURL>`, see Printing Deployment Details.

2. In the OAA Administration UI console, click the Application Navigation hamburger menu on the top left.
3. From the menu, click **OAAM Policy Explorer**. The **OAAM Policies** windows opens. This page shows a list of all your OAAM policies that are transitioned.
4. Click on the required policy.
5. You can activate or deactivate the policy by clicking on the **Activate** or **Deactive** button.

You can also update the transitioned OAAM policies using the Policy Browser REST APIs. For details, see Policy Browser REST Endpoints.

Part V
Managing Oracle Advanced Authentication

11 Configuring Oracle Advanced Authentication

Topics

11.1 Creating Integration Agents in OAA

You must create an integration agent to integrate client applications with OAA. You can create OAM integration Agents and Oracle RADIUS Integration Agents. You can also create Other Integration Agents for use with your own REST API client applications.

You can create integration agents either using REST APIs or OAA Administration UI console. For details about creating integration agents using REST APIs, see REST API for Administration in Oracle Advanced Authentication.

To create an Oracle RADIUS Integration Agent, see: Use Oracle RADIUS Agent with Oracle Advanced Authentication for Multi-Factor Authentication

To create an OAM integration agent in the OAA Administration UI console:

Note:

For full details on integrating OAM with OAA, see Integrate Oracle Access Management with Oracle Advanced Authentication

1. Login to the OAA Administration console `https://<AdminUrl>`. You are redirected to the OAM login page as the console is protected by OAM OAuth. Specify your credentials and login.
2. Under **Quick Actions** select **Create OAM Integration Agent**.
3. In the **Create Integration Agent** window, specify the following:

   a. **Name**: For OAM integration, the value must be the same as the partner name created while registering OAA as TAP partner. For more information, see Register OAA as a TAP Partner in OAM
   b. **Description**: Add a description about the integration agent.
   c. **Integration Agent Type**: **Oracle Access Management** is selected by default.
   d. **Client ID**: Click **Re-Generate** to create a Client ID and then click **Copy** to copy the generated Client ID.

      Note:

      The Client ID needs to be provided when configuring OAM for integration with OAA using the **OAAAuthnPlugin**. For more information, see Install and configure the OAA Plugin in OAM

   e. **Client Secret**: Click **Re-Generate** to create a Client Secret and then Click **Copy** to copy the generated Client Secret.

      Note:

      The Client Secret needs to be provided when configuring OAM for integration with OAA using the **OAAAuthnPlugin**

   f. **Private Key File**: Drag and Drop the Java KeyStore file (.jks) that was created after registering OAM as a TAP partner of OAA. For example, **OAMOAAKeyStore.jks**.
   g. **Private key Password**: Specify the password that you had provided while registering OAM as a TAP partner of OAA.
4. Click **Save**

To create an integration agent for use with your own REST API client applications:

1. Login to the OAA Administration console `https://<AdminUrl>`. You are redirected to the OAM login page as the console is protected by OAM OAuth. Specify your credentials and login.
2. Under **Quick Actions** select **Create Other Integration Agent**.
3. In the **Create Integration Agent** window, specify the following:
   a. **Name**: Enter a name for your integration agent.
   b. **Description**: Add a description about the integration agent.
   c. **Integration Agent Type**: **API** is selected by default.
   d. **Client ID**: Click **Re-Generate** to create a Client ID and then click **Copy** to copy the generated Client ID.

      Note:

      The Client ID needs to be provided when configuring your application.

e. **Client Secret**: Click **Re-Generate** to create a Client Secret and then Click **Copy** to copy the generated Client Secret.

> **Note:**
>
> The Client Secret needs to be provided when configuring your application.

4. Click **Save**.

11.2 Creating Assurance Levels in OAA

You can create assurance levels for an integration agent either using REST APIs or the OAA Administration UI console. For more details about using REST APIs, see Create an Assurance Level.

The following steps provide instructions to create an assurance level for an integration agent on the OAA Administration UI console:

1. Login to the OAA Administration console `https://<AdminUrl>`. You are redirected to the OAM login page as the console is protected by OAM OAuth. Specify your credentials and login.
2. If the integration agent has been recently created, it is shown under **Recent Activity**. However, if the integration agent does not appear under **Recent Activity**, do one of the following:
   - Click **Show more agents**
   - Click the **Application Navigation** icon on the top-left of the page and select **Manage Integration Agents**
3. In the **Integration Agents** window, select the integration agent for which you need to create the assurance level.
4. Under the **Assurance Levels** tab, click **Create**.
5. Specify the required details:
   - **Name**: Specify the name for this assurance level
   - **Description**: Provide the description for the assurance level.
6. Click **Create**.

11.3 Configuring Rules for an Assurance Level in OAA

You can manage rules for an assurance level using the OAA Administration UI console or REST APIs. If you create rules for an assurance level in the OAA Administration UI console, a policy for those rules is automatically created for you. If using REST API's to create rules then you must create the policy first using the REST API. For more details about using REST APIs to create a policy and associated rules, see Create Policy.

The following steps provide instructions to create rules for an assurance level on the OAA Administration UI console:

1. Login to the OAA Administration console `https://<AdminUrl>`. You are redirected to the OAM login page as the console is protected by OAM OAuth. Specify your credentials and login.
2. If the integration agent has been recently created, it is shown under **Recent Activity**. However, if the integration agent does not appear under **Recent Activity**, do one of the following:
   - Click **Show more agents**
   - Click the **Application Navigation** icon on the top-left of the page and select **Manage Integration Agents**
3. In the **Integration Agents** window, select the required integration agent.
4. Under the **Assurance Levels** tab, select the required assurance level for which you are required to define rules
5. Under **Uses** select the required factors to assign to the assurance level. For example, select **Oracle Mobile Authenticator**, **Email Challenge** and **SMS Challenge**.
6. Under **If the following condition(s) are met**, select the **Attribute Name**, **Operator**, and **Values** to create rules. Based on the Attribute Name selected, corresponding options appear in the Operator drop-down and Values fields. For example, for `User In Group` with operator `Contains Any` specify the value in the **Values** field. For `User In Group` with operator **In Group**, the values field changes to **Group**, and you can select a group name from the drop-down.
   The following options are supported in Attribute Name:
   - User in Group
   - User's Group
   - User Login
   - User Attributes
   - Current Authentication Level
   - IP Address
   - Application ID
   - Parameter
   - Resource URL
   - New Authentication Level
   - Agent
   - IP Address X-Forwarded-For
7. Click **Validate Rule**.
8. Click **Save**.
9. Create additional rules, if necessary, by clicking the **+** icon.

11.4 Creating Groups in OAA

You can create groups for an integration agent in OAA either using REST APIs or OAA Administration UI console. For more details about using REST APIs, see Create Groups.

The following steps provide instructions to create a group for an integration agent in the OAA Administration UI console:

1. Login to the OAA Administration console `https://<AdminUrl>`. You are redirected to the OAM login page as the console is protected by OAM OAuth. Specify your credentials and login.
2. If the integration agent has been recently created, it is shown under **Recent Activity**. However, if the integration agent does not appear under **Recent Activity**, do one of the following:
   - Click **Show more agents**
   - Click the **Application Navigation** icon on the top-left of the page and select **Manage Integration Agents**
3. In the **Integration Agents** window, select the integration agent, for which you are required to create a group.
4. Under the **Groups** tab, click **Create**.
5. Specify the required details:
   a. **Name**: Specify the name of the group.
   b. **Description**: Specify a description for the group.
   c. **Type**: From the drop-down, select the required type.
   d. Click **Create** .
   e. Under **Values** tab, click **Add**. Add the corresponding value for the type selected in the previous step. See the following table for details.
6. Click **Save**.

Table 11-1 Type Value Reference for Groups

Type	Description and Values
User ID	Select this to create a group based on the user id. Specify the user ID in the values field.
IP Address	Select this to create a group based on the IP address. Specify the IP address in the values field.

Table 11-1 (Cont.) Type Value Reference for Groups

Type	Description and Values
IP Address Range	Select this to create a group based on the value lying between the specified IP Range: 1. **Name**: Specify a name. 2. **Description**: Specify a description. 3. **From**: Specify the starting IP address of the range. 4. **To**: Specify the ending IP address of the range.
String	Select this to create a group based on any specify value required. Specify the string in the values field.

11.5 Registering Users with Challenge Factors in OAA

OAA provides REST APIs for registering users with specific challenge factors.

Use the `<OAAService>/preferences/v1` REST API to register the necessary challenge factors for users.

For details about finding the `<OAAService>` URL and authenticating, see OAA Runtime API.

For details about the Preferences REST Endpoint, see REST API for OAA Runtime. These factors can be further managed by users in the User Preferences UI. For more information, see Managing User Preferences UI.

Example 1: Sample Request to Register User with Oracle Mobile Authenticator (OMA)

The following sample request shows how to register a user `testuser1` with groupID `UserGroup1` with OMA:

```
curl --location -g --request POST '<OAAService>/preferences/v1' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic <Base64Encoded(<username>:<password>)>'
\
--data '{
    "userId": "testUser1",
    "groupId": "UserGroup1",
  "factorsRegistered": [
            {
              "factorAttributes": [
                {
                  "factorAttributeValue": [
                    {
                      "value": "omasecretvalue1",
                      "name": "Device1",
                      "isEnabled": true
                    }
                  ],
                  "factorAttributeName": "omatotpsecretkey"
                }
              ],
              "factorKey": "ChallengeOMATOTP",
              "isPreferred": false,
              "isValidated": true
            }
          ]
        }'
```

Note:

The `value` for `factorAttribueValue` supports alphanumerics only.

Example 2: Sample Request to Register User with Email

The following sample request shows how to register a user `testuser1` with groupID `UserGroup1` with Email:

```
curl --location -g --request POST '<OAAService>/preferences/v1' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic <Base64Encoded(<username>:<password>)>' \
--data '{
    "userId": "testUser1",
    "groupId": "UserGroup1",
  "factorsRegistered": [
          {
            "factorAttributes": [
              {
                "factorAttributeValue": [
                  {
                    "value": "test.user@example.com",
                    "name": "Device1",
                    "isEnabled": true
                  }
                ],
                "factorAttributeName": "email"
              }
            ],
            "factorKey": "ChallengeEmail",
            "isPreferred": false,
            "isValidated": true
          },
```

Once a user is registered they can manage their factors using the User Preferences UI. See, Managing Factors in the User Preferences UI.

Note:

The parameter `isValidated` is `true` by default. If using factor verification as per Configuring Factor Verification, if `isValidated` is `true` then the user will be registered as verified, whereas if the value is set to `false`, the user will be registered as unverified.

11.6 Managing Factors in the User Preferences UI

OAA provides users with a User Preferences UI for managing factors.

Access the User Preferences UI by launching a browser and accessing `https://<SpuiURL>`. The user logs in to the console using their username (e.g `testuser1`) and their password set in the OAM OAuth identity store.

Note:

For details on finding the `<SpuiUrl>`, see Printing Deployment Details.

On the User Preferences Page, for each of the factors registered, a corresponding challenge factor tile is displayed. For example, if the user is registered with Oracle Mobile Authenticator (OMA) and Email challenge factors, the corresponding tiles named **Oracle Mobile Authenticator** and **Email Challenge** are displayed.

On the factor tiles, you can choose to Disable, Enable, Delete or set your Preferred Authentication factor. Click on the **...** menu button on the factor tiles and select one of the following options:

- **Disable**: This factor is disabled, and will not be displayed during the second-factor authentication.
- **Preferred Authentication Factor**: This factor is set as the default challenge factor and is displayed automatically during the second-factor authentication. A green dot appears on the default factor in the User Preferences UI page.
- **Delete**: Deletes the registered factor.
- **Enable**: If the factor is disabled, you can choose to enable it by selecting this option.

In addition to the registered factors, you can also add more factors for second-factor authentication. Click **Add Authentication Factor** and choose the required factor from the displayed list. Based on the factors registered for the user, one, some, or all of the following factors are displayed:

Factors	Values
Oracle Mobile Authenticator	**Friendly Name**: Specify a name. **Key**: A key is generated by OAA. **QR Code**: Scan the QR code using the Oracle Mobile Authenticator, Google Authenticator, or Microsoft Authenticator application.

Factors	Values
Email Challenge	**Friendly Name**: Specify a name. **Email**: Specify the required email.
FIDO2 Challenge	**Friendly Name**: Specify a name. Click Register and press the button on your FIDO2 device. The key is copied into OAA.
OMA Push Notification Challenge	Scan the QR code, or manually register your device. In the OMA application enter the userid and PIN displayed here.
Security Question Challenge	**Question 1**: Select a question to answer. **Answer 1**: Provide an answer the question. Repeat for the remaining Question and Answers.
SMS Challenge	**Friendly Name**: Specify a name. **Phone**: Specify the phone number.
Yubico OTP Challenge	**Friendly Name**: Specify a name. **Public ID**: Type the Public ID. It must be the same as the Public ID (or serial) specified while configuring the Yubico OTP for your YubiKey device. **Secret Key**: Type the Secret Key. It must be the same as the Secret Key generated while configuring the Yubico OTP for your YubiKey device. **Private ID**: Type the Private ID. It must be the same as the Private ID generated while configuring the Yubico OTP for your YubiKey device.

For more details on configuring Mobile Authenticator, Security Question Challenge, Yubikey OTP Challenge, and FIDO2 Challenge, see the following tutorials:

- Configuring Mobile Authenticator TOTP with Oracle Advanced Authentication
- Configuring Knowledge Based Authentication.
- Configuring FIDO2 with Oracle Advanced Authentication.
- Configuring YubiKey with Oracle Advanced Authentication.

11.7 Configuring Oracle UMS Server for Email and SMS

OAA supports Oracle UMS out-of-the-box for providing email and SMS challenges

To integrate Oracle UMS with OAA for providing email and SMS challenge factors, use the `<PolicyUrl>/policy/config/property/v1` REST API as shown in the following sample request.

Note:

In this case remove `/oaa-policy` from the `<PolicyUrl>`, for example use `https://<host>:<port>/policy/config/property/v1` not `https://<host>:<port>/oaa-policy/policy/config/property/v1`

For details about finding the `PolicyUrl` and authenticating, see OAA Admin API.

For details about the Configuration Properties REST API, see Configuration Properties REST Endpoints.

Sample Request

```
curl --location -g --request PUT '<PolicyUrl>/policy/config/
property/v1' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic <Base64Encoded(<username>:<password>)>'
\--data '[
                {
        "name":
"bharosa.uio.default.challenge.type.enum.ChallengeEmail.umsClientURL",
        "value": "<UMS_SERVER_URL>"
},
{
        "name":
"bharosa.uio.default.challenge.type.enum.ChallengeEmail.umsClientName",
        "value": "<UMS_ADMIN_USER>"
},
{
        "name":
"bharosa.uio.default.challenge.type.enum.ChallengeEmail.umsClientPass",
        "value": "<UMS_ADMIN_PASSWORD>"
},
{        "name":
"bharosa.uio.default.challenge.type.enum.ChallengeEmail.fromAddress",
        "value": "<fromAddress>"
},
{
        "name":
"bharosa.uio.default.challenge.type.enum.ChallengeSMS.umsClientURL",
        "value": "<UMS_SERVER_URL>"
},
{
        "name":
"bharosa.uio.default.challenge.type.enum.ChallengeSMS.umsClientName",
        "value": "<UMS_ADMIN_USER>"
},
{
        "name":
"bharosa.uio.default.challenge.type.enum.ChallengeSMS.umsClientPass",
        "value": "<UMS_ADMIN_PASSWORD>"
}
]'
```

where:

- `<UMS_SERVER_URL>` is the UMS server URL, for example: `http://ums.example.com:8001/ucs/messaging/webservice`.
- `<UMS_ADMIN_USER>` is the username for the UMS service.
- `<UMS_ADMIN_PASSWORD>` is the corresponding password for `<ums_username>`.
- `<fromAddress>` is the email address from which end users will receive the One Time Password (OTP), for example `oaa@example.com`

For implementing your own email and SMS servers, see Customizing Email and SMS Messaging Provider.

11.8 Configuration Properties for OAA

OAA provides REST APIs for configuring properties for challenge factors and other settings.

Use the `<PolicyUrl>/policy/config/property/v1` REST API to configure properties.

Note:

In this case remove `/oaa-policy` from the `<PolicyUrl>`, for example use `https://<host>:<port>/policy/config/property/v1` not `https://<host>:<port>/oaa-policy/policy/config/property/v1`

For details about finding the `PolicyUrl` and authenticating, see OAA Admin API.

For details about the Configuration Properties REST Endpoint, see Configuration Properties REST Endpoints.

Table 11-2 Configuration Properties for OAA

Property Name	Default Value	Description
`bharosa.uio.default.all.factor.challengecounter.expiryTime`	`1800000` milliseconds	Expiry time of the challenge counter lock for the factors. This is the time duration for which the challenge remains unavailable for users, after challenge is locked due to maximum number of unsuccessful retries.
`bharosa.uio.default.all.factor.retry.count`	`10`	Maximum number of unsuccessful retries of the challenge for the factors. Beyond this count the challenge is locked.
`bharosa.uio.default.challenge.type.enum.ChallengeEmail.appName`	`OAA`	Name of the application.
`bharosa.uio.default.challenge.type.enum.ChallengeEmail.challengeText`	`Enter OTP sent to {0}.`	Prompt message to enter One Time Pin (OTP) on the end-user challenge page.

Table 11-2 (Cont.) Configuration Properties for OAA

Property Name	Default Value	Description
`bharosa.uio.default.challenge.type.enum.ChallengeEmail.fromAddress`	`oaa@oracle.com`	Email address of the email sending entity.
`bharosa.uio.default.challenge.type.enum.ChallengeEmail.fromName`	`OAA`	Name of the From email sending entity.
`bharosa.uio.default.challenge.type.enum.ChallengeEmail.msgIPTemplate`	`IP Address:`	Part of the email template to display message IP addres.s
`bharosa.uio.default.challenge.type.enum.ChallengeEmail.msgPinTemplate`	`Please use following one time pin to login to protected resource:`	Part of the email template to display One Time Pin (OTP).
`bharosa.uio.default.challenge.type.enum.ChallengeEmail.msgResourceURLTemplate`	`Resource URL Access:`	Part of the email template to display message resource URL.
`bharosa.uio.default.challenge.type.enum.ChallengeEmail.msgSubject`	`One Time Pin: OAA`	Subject title of the email template.
`bharosa.uio.default.challenge.type.enum.ChallengeEmail.challengeCounterExpiryTime`	`1800000` milliseconds	Expiry time of the challenge counter lock. This is the time duration for which the challenge remains unavailable for users, after challenge is locked due to maximum number of unsuccessful retries. If the value is not provided then the value for `bharosa.uio.default.all.factor.challengecounter.expiryTime` is used (default is 1800000 milliseconds) .
`bharosa.uio.default.challenge.type.enum.ChallengeEmail.retrycount`		Maximum number of unsuccessful retries of the challenge. Beyond this count the challenge is locked. If the value is not provided then the value for `bharosa.uio.default.all.factor.retry.count` is used (default is 10).
`bharosa.uio.default.challenge.type.enum.ChallengeEmail.msgTimeTemplate`	`Time of Access:`	Part of the email template to display message time.
`bharosa.uio.default.challenge.type.enum.ChallengeEmail.promptmessage`	`Send OTP to {0}`	Prompt message to send One Time Pin (OTP) through email used on end-user challenge page.

Table 11-2 (Cont.) Configuration Properties for OAA

Property Name	Default Value	Description
`bharosa.uio.default.challenge.type.enum.ChallengeEmail.promptselectmessage`	`Please select one of following addresses to receive OTP.`	Prompt message to select addresses to send One Time Pin (OTP) to user on end-user challenge page.
`bharosa.uio.default.challenge.type.enum.ChallengeOMATOTP.challengeText`	`Enter OTP from device {1}`	Prompt message to enter time-based One Time Pin (OTP) on end-user challenge page.
`bharosa.uio.default.challenge.type.enum.ChallengeOMATOTP.promptselectmessage`	`Please select one of following channels`	Prompt message to select channels to send time-based One Time Pin (OTP) to user on end-user challenge page.
`bharosa.uio.default.challenge.type.enum.ChallengeOMATOTP.challengeCounterExpiryTime`	`1800000` milliseconds	Expiry time of the challenge counter lock. This is the time duration for which the challenge remains unavailable for users, after challenge is locked due to maximum number of unsuccessful retries. If the value is not provided then the value for `bharosa.uio.default.all.factor.challengecounter.expiryTime` is used (default is 1800000 milliseconds).
`bharosa.uio.default.challenge.type.enum.ChallengeOMATOTP.retrycount`		Maximum number of unsuccessful retries of the challenge. Beyond this count the challenge is locked. If the value is not provided then the value specified for `bharosa.uio.default.all.factor.retry.count` is used (default is 10).
`bharosa.uio.default.challenge.type.enum.ChallengeOMATOTP.registration.showSecretKeyText`	`true`	Displays a secret key in the User Preferences UI, for use with Oracle Mobile Authenticator, Google Authenticator, or Microsoft Authenticator. If the value is set to `false`, the secret key isn't displayed.
`bharosa.uio.default.challenge.type.enum.ChallengeOMATOTP.registration.showQrcode`	`true`	Displays a QR code in the User Preferences UI, for use with Oracle Mobile Authenticator, Google Authenticator, or Microsoft Authenticator. If the value is set to `false`, the QR code isn't displayed.

Table 11-2 (Cont.) Configuration Properties for OAA

Property Name	Default Value	Description
`bharosa.uio.default.challenge.type.enum.ChallengeOMATOTP.keyExpiryEnabled`	`false`	A boolean value that indicates whether or not secret key expiration is enabled. When enabled, the Time-based One Time Password (TOTP) secret key expiration time is checked during the challenge flow. If the key has expired, the challenge flow fails and the key is deleted. If the key has not expired, the challenge flow will continue as usual.
`bharosa.uio.default.challenge.type.enum.ChallengeOMATOTP.keyExpiryTimeMinutes`	`60` minutes	Specifies the key's expiration time in minutes. This must be a positive whole number.
`bharosa.uio.default.challenge.type.enum.ChallengeSMS.challengeCounterExpiryTime`	`1800000` milliseconds	Expiry time of the challenge counter lock. This is the time duration for which the challenge remains unavailable for users, after challenge is locked due to maximum number of unsuccessful retries. If the value is not provided then the value for `bharosa.uio.default.all.factor.challengecounter.expiryTime` is used (default is 1800000 milliseconds).
`bharosa.uio.default.challenge.type.enum.ChallengeSMS.retrycount`		Maximum number of unsuccessful retries of the challenge. Beyond this count the challenge is locked. If the value is not provided then the value specified for `bharosa.uio.default.all.factor.retry.count` is used (default is 10).
`bharosa.uio.default.challenge.type.enum.ChallengeSMS.appName`	`OAA`	Name of the application.
`bharosa.uio.default.challenge.type.enum.ChallengeSMS.challengeText`	`Enter OTP sent to {0}.`	Prompt message to enter One Time Pin (OTP) on end-user challenge page.
`bharosa.uio.default.challenge.type.enum.ChallengeSMS.fromAddress`	`oaa@oracle.com`	Mobile number of the SMS sending entity.
`bharosa.uio.default.challenge.type.enum.ChallengeSMS.fromName`	`OAA`	Name of the From SMS sending entity.

Table 11-2 (Cont.) Configuration Properties for OAA

Property Name	Default Value	Description
`bharosa.uio.default.challenge.type.enum.ChallengeSMS.msgIPTemplate`	`IP Address:`	Part of the SMS template to display message IP address.
`bharosa.uio.default.challenge.type.enum.ChallengeSMS.msgPinTemplate`	`Please use following one time pin to login to protected resource:`	Part of the SMS template to display One Time Pin (OTP).
`bharosa.uio.default.challenge.type.enum.ChallengeSMS.msgResourceURLTemplate`	`Resource URL Access:`	Part of the SMS template to display message resource URL.
`bharosa.uio.default.challenge.type.enum.ChallengeSMS.msgSubject`	`One Time Pin: OAA`	Subject title of the SMS template.
`bharosa.uio.default.challenge.type.enum.ChallengeSMS.msgTimeTemplate`	`Time of Access:`	Part of the SMS template to display message time.
`bharosa.uio.default.challenge.type.enum.ChallengeSMS.promptmessage`	`Send OTP to phone {0}`	Prompt message to send One Time Pin (OTP) through SMS used on end-user challenge page.
`bharosa.uio.default.challenge.type.enum.ChallengeSMS.promptselectmessage`	`Please select one of following numbers to receive OTP.`	Prompt message to select addresses to send One Time Pin (OTP) to user on end-user challenge page.
`bharosa.uio.default.challenge.type.enum.ChallengeTOTP.promptmessage`	`Enter OTP from registered phone`	Prompt message to send time-based One Time Pin (OTP) used on end-user challenge page.
`bharosa.uio.default.challenge.type.enum.ChallengeYubicoOTP.challengeCounterExpiryTime`	`1800000` milliseconds	Expiry time of the challenge counter lock. This is the time duration for which the challenge remains unavailable for users, after challenge is locked due to maximum number of unsuccessful retries. If the value is not provided then the value for `bharosa.uio.default.all.factor.challengecounter.expiryTime` is used (default is 1800000 milliseconds).
`bharosa.uio.default.challenge.type.enum.ChallengeYubicoOTP.retrycount`		Maximum number of unsuccessful retries of the challenge. Beyond this count the challenge is locked. If the value is not provided then the value specified for `bharosa.uio.default.all.factor.retry.count` is used (default is 10).

Table 11-2 (Cont.) Configuration Properties for OAA

Property Name	Default Value	Description
`bharosa.uio.default.challenge.type.enum.ChallengeFIDO2.challengeCounterExpiryTime`	`1800000` milliseconds	Expiry time of the challenge counter lock. This is the time duration for which the challenge remains unavailable for users, after challenge is locked due to maximum number of unsuccessful retries. If the value is not provided then the value for `bharosa.uio.default.all.factor.challengecounter.expiryTime` is used (default is 1800000 milliseconds).
`bharosa.uio.default.challenge.type.enum.ChallengeFIDO2.retrycount`		Maximum number of unsuccessful retries of the challenge. Beyond this count the challenge is locked. If the value is not provided then the value specified for `bharosa.uio.default.all.factor.retry.count` is used (default is 10).
`oracle.security.oaa.kba.challenge.number`	`1`	Number of security questions that the user will be asked to answer during the challenge flow. This should be set to a value no larger than the maximum number of active questions answered by the user during security question registration. **Note**: This property should be used in conjunction with the `oracle.security.oaa.kba.challenge.separator` property described in the row below.

Table 11-2 (Cont.) Configuration Properties for OAA

Property Name	Default Value	Description
`oracle.security.oaa.kba.challenge.separator`	|	If `oracle.security.oaa.kba.challenge.number` is set to a value greater than 1, the generated challenge will contain the multiple challenges as a string, separated by the value of `oracle.security.oaa.kba.challenge.separator`. For example: `What is your name?|What is your age?|What is your birthplace?`. When the response to the challenge is presented to the OAA server, the response is also expected to be seperated by the same separator. By default the value is "|". If you anticipate any of the questions or answers could contain the value "|" then you must change this parameter to use a seperator that is is not contained in the question or answer. To override this value set `oracle.security.oaa.kba.challenge.separator` to a character or combination of characters of your choice. **Note**: Changing the separator may impact in flight KBA authentications, Hence, perform updates to this configuration when the KBA service is offline.
`oaa.user.auth.question.authn.counter.enabled`	`true`	If this property is `true`, the risk counters are incremented.
`oaa.user.auth.question.next.seq`	`false`	If this property is `false`, `oaam.kba.questions.randomorder` is `true`, and `oracle.security.oaa.kba.challenge.number` is 1, the questions selected from picklist are at random. Else, the user is challenged by questions from the picklist in sequential order.

Table 11-2 (Cont.) Configuration Properties for OAA

Property Name	Default Value	Description
`oaam.kba.questions.randomorder`	`false`	If this property is `true`, `oaa.user.auth.question.next.seq` is `false`, and `oracle.security.oaa.kba.challenge.number` is 1, questions selected from picklist are at random. Else, the user is challenged by questions from the picklist in sequential order.
`bharosa.kba.questions.trim.answers.for.matching`	`true`	If this property is set to `true`, the answer and the matched value are trimmed before matching.
`oaa.browser.cookie.domain`		In case of an OAA-OARM install this must be set to the OAA host domain to collect the device cookie properly. For example, if the OAA is accessible on `https://oaa.example.com`, then set the value to `oaa.example.com`.
`oaa.risk.integration.postauth.cp`	`postauth`	Defines the default risk assurance level for OAA assurance level. The default value is `postauth` and should not be changed. **Note:** This property is related to OAA-OARM integration.
`oaa.policy.assurance.level.default.action`	`Challenge`	Defines the default action associated with the OAA assurance level. **Note:** This property is related to OAA-OARM integration.
`profile.type.enum.<AssuranceLevelKey>.riskcheckpoint`		Checkpoint associated with the existing assurance level. **Note:** This property is related to OAA-OARM integration.

Table 11-2 (Cont.) Configuration Properties for OAA

Property Name	Default Value	Description

`profile.type.enum.<AssuranceLevelKey>.defaultaction`		Default action associated with the existing assurance level. Acceptable values are `Allow`, `Block`, and `Challenge`. For instance:
		
		`[`
		`{`
		`        "name": "profile.type.enum.ChallengeMFA.defaultaction",`
		`        "value": "<Allow/Block/Challenge>",`
		`        "source": "database"`
		` }`
		`]`
		
		Note: This property is related to OAA-OARM integration.

`rule.action.enum.<actionName>.priority`		Defines the priority of the action. It can be a integer value or string "max" to identify the highest priority. For instance:
		
		`[`
		`    {`
		`        "name": "rule.action.enum.Block.priority",`
		`        "value": "max",`
		`        "source": "database"`
		`    }`
		`]`
		
		Note: This property is related to OAA-OARM integration.

To configure properties to customize the user interface (UI) for the OAA Administration Console, User Preferences Console, and Runtime UI, see Customizing the OAA User Interface.

To configure properties for Factor Verification, see Configuring Factor Verification.

11.9 Configuring Factor Verification

OAA allows you to configure factor verification. Factor verification allows users to verify a factor in the User Preferences UI after the factor has been added. This allows a user check the factor is working before it is used in a user challenge. By default, factor verification is disabled.

Topics

The following topics describe how to configure factor verification:

- Creating a Verification Integration Agent
- Creating an Assurance Level for the Verification Integration Agent
- Configuring Properties for Factor Verification
- Testing Factor Verification

11.9.1 Creating a Verification Integration Agent

To enable factor verification, you must create a verification integration agent.

You can create integration agents either using REST APIs or OAA Administration UI console. For details about creating integration agents using REST APIs, see REST API for Administration in Oracle Advanced Authentication.

To create a verification integration agent:

1. Login to the OAA Administration console `https://<AdminUrl>`. You are redirected to the OAM login page as the console is protected by OAM OAuth. Specify your credentials and login.
2. Under **Quick Actions** select **Create Other Integration Agent**.
3. In the **Create Integration Agent** window, specify the following:
   a. **Name**: Enter a name for your integration agent, for example `VerificationFlowAgent`.

      Note:

      The property `oaa.default.spui.pref.runtime.verification.agentId` is set to `VerificationFlowAgent` by default. If you choose to give your agent a different name then you must configure the property to match. See Configuring Properties for Factor Verification.

   b. **Description**: Add a description for the integration agent.
   c. **Integration Agent Type**: **API** is selected by default.
   d. Click **Save**.

Next steps: Creating an Assurance Level for the Verification Integration Agent.

11.9.2 Creating an Assurance Level for the Verification Integration Agent

Create an assurance level for the verification integration agent.

You can create assurance levels either using REST APIs or OAA Administration UI console. For details about creating integration agents using REST APIs, see REST API for Administration in Oracle Advanced Authentication.

To create an assurance level for the verification integration agent:

1. In the **Integration Agents** window, select the verification integration agent for which you need to create the assurance level.
2. Under the **Assurance Levels** tab, click **Create**.
3. Specify the required details:
   a. **Name**: Specify the name for this assurance level, for example `FactorVerificationAL`.

      Note:

      The property `oaa.default.spui.pref.runtime.verification.assuranceLevel` is set to `FactorVerificationAL` by default. If you choose to give your assurance level a different name then you must configure the property to match. See Configuring Properties for Factor Verification

   b. **Description**: Provide the description for the assurance level.
   c. Click **Create**.
   d. Click the Assurance Level created.
   e. Under **Uses** select the factors for which you want to configure factor verification.

      Note:

      Factor verification is only supported for Oracle Mobile Authenticator, Email Challenge, Yubico OTP Challenge, and SMS Challenge.

4. Click **Save**.

Next steps: Configuring Properties for Factor Verification.

11.9.3 Configuring Properties for Factor Verification

To enable factor verification you must set configuration properties.

The following table lists the OAA properties that you must configure to enable factor verification.

Table 11-3 Factor Verification Properties

Property Name	Description	Default Value
`oaa.default.spui.pref.runtime.verification.enabled`	This property determines if factor verification is enabled or disabled. To enable factor verification set this value to `true`	`false`
`oaa.default.spui.pref.runtime.verification.agentId`	The name of the verification integration agent. If you create a verification agent with a name other than the default `VerificationFlowAgent`, you must set this property to the name of the agent created.	`VerificationFlowAgent`
`oaa.default.spui.pref.runtime.verification.assuranceLevel`	The name of the assurance level for the verification agent. If you create an assurance level with a name other than the default `FactorVerificationAL`, you must set this property to the name of the assurance level created.	`FactorVerificationAL`

Use the `<PolicyUrl>/policy/config/property/v1` REST API to configure properties.

Note:

In this case remove `/oaa-policy` from the `<PolicyUrl>`, for example use `https://<host>:<port>/policy/config/property/v1` not `https://<host>:<port>/oaa-policy/policy/config/property/v1`

For details about finding the `PolicyUrl` and authenticating, see OAA Admin API.

For details about the Configuration Properties REST Endpoint, see Configuration Properties REST Endpoints.

Next steps: Testing Factor Verification.

11.9.4 Testing Factor Verification

To test factor verification:

1. Access the User Preferences UI by launching a browser and accessing `https://<SpuiURL>`. The user logs in to the console using their username and password set in the OAM OAuth identity store.

   Note:

   For details on finding the `<SpuiUrl>`, see Printing Deployment Details.

2. Under **Authentication Factors** select **Add Authentication Factor** and select an authentication factor. In this example **Email Challenge** is selected.
3. In the **Setup Security Code via Email** page, enter a **Friendly Name** and **Email address**. As factor verification is enabled, two new options are shown: **Verify Now** and **Verify Later**.

If you select **Verify Now** you will be asked to enter the verification code. In this example the verification code will be sent to the email address. Enter the verification code from the email and select **Verify and Save**. If verification is successful you will be returned to the **Authentication Factors** screen and the authentication factor will show as **Enabled**.

If you select **Verify Later** you will be returned to the **Authentication Factors** screen. The factor added will show as **Unverified**.

Note:

If **Verify Later** is selected, the factor added will not be presented in a user challenge until it is verified.

If **Verify Later** is selected, the factor is saved as **Unverified**. It can verified by selecting **Verify** from the factor drop down menu on the **Authentication Factors** screen. Once the factor is verified it will show as **Enabled**.

Note:

Any factors added prior to enabling factor verification will show either **Enabled** or **Disabled** and will not need to go through verification.

Important Note for Upgrades

If you are upgrading from a previous release where factor verfication wasn't supported, all previously registered factors for a user will automatically be verified after upgrade. This is true for all previously enabled and disabled factors for a user. Any new factors registered for the user after upgrade, will use factor verification.

11.10 Configuring Security Questions for Knowledge-Based Authentication

Knowledge-based authentication (KBA) is an authentication method which is used to challenge the user to prove identity based on the user’s answers substantiated by a real-time interactive question and answer process.

The KBA feature provides a rich set of challenge questions, logic behind presenting these challenge questions to users, and validations to control the answers that users can provide.

KBA is a secondary authentication feature, which is presented to the user after successful primary authentication (for example, a user entering user name and password) to enhance the security.

KBA provides an infrastructure for:

- **Questions:** Users to select challenge questions and provide answers which are used to challenge them later on.
- **Categories:** Manages the question categories in the system.
- **Registration Logic:** Manages the level of algorithm logic used for the registration for challenge questions and answers.
- **Answer Logic:** To intelligently detect the correct answers in the challenge response process.
- **Validations:** Manages the validation for the answers given by a user at the time of registration.

This chapter introduces you to the key concepts behind KBA. It contains the following topics:

- About KBA Registration
- Configuring Registration Logic
- Configuring Answer Logic
- About Top Categories
- About Top Questions
- About Disabling Question and Category Logic
- About Deleting Question and Category Logic
- Configuring Validations for Answer Registration

11.10.1 About KBA Registration

During registration, which could be enrollment, opening a new account, or other events such as a reset, the user is asked to select questions and provide answers. The order of questions that are presented to a user during the registration phase is random using configurable parameters.

The challenge questions selected at registration or during a reset are then used for authenticating users using a challenge/response process where users are challenged with one or more questions to provide identity before they are allowed to proceed with high risk log ins or access transactions.

11.10.2 Configuring Registration Logic

Registration Logic manages the registration of challenge questions and answers.

The KBA feature offers a large framework of questions for obtaining answers from the user during registration or reset. During KBA registration the user is presented with a Question Set, which is a subset of the challenge questions library. The Question Set allotted to the user is generated based on the settings configured in the Registration Logic. This Question Set prevents an imposter from harvesting questions for use in a phishing exercise.

The Question Set is broken down into several drop-down lists that contain questions to select from. The drop-down lists with questions is called a "menu."

To configure the Registration Logic, you specify the settings for Question Set generation as follows:

- The number of questions to be registered
- The number of questions per menu
- The number of categories per menu

You can configure the number of questions that a user must register, the number of questions that appear in each menu, and the number of categories per menu. As standard, questions are grouped into categories. The user is allowed to select one question from each menu and enter answers for them. Only one question from each question menu can be registered.

Let us consider the following example to see how the Registration Logic settings affect the Question Set of a customer.

Question/Menu	Categories/Menu	Questions/Category in a Menu
7	4	2+2+2+1

The example results in registration menus containing 2 questions from category A, and 2 questions from category B, and 2 questions from category C, and 1 question from category D. This continues in a round robin fashion as needed. If there are any categories with an insufficient number of questions or an insufficient number of categories duplicate questions can result.

For example to generate a Question Set with:

- 3 menus
- 5 questions per menu
- 5 categories per menu

The algorithm tries to pick one question each from 15 categories if 15 categories are available. The minimum number of questions per category should be equal to the number of questions in the Question Set divided by the total number of categories.

Pre-requisite for Configuring Registration Logic for Locales

The Administrator must ensure that there are enough questions in the database for each of the supported locale as configured in OAA Admin during deployment; otherwise, the application displays only the English language questions during registration.

The number of locale-specific questions must be equal to or greater than the "Questions User Will Register" multiplied by the "Questions per Menu" multiplied by the "Categories per Menu."

11.10.3 Configuring Answer Logic

Answer Logic validates if the answer provided by the user matches with what was provided during registration.

Answer Logic, a feature of KBA, increases the usability of challenge questions. Administrators can adjust how exact the challenge answers given by end users must match the answers they gave at the time of registration. If the answer given by a user is fundamentally correct but there are minor variations such as typos, misspellings, and abbreviations they should pass. The increased usability of KBA reduces or eliminates the need for unnecessary call center involvement in moderate risk situations and self service flows.

Answer Logic consists of advanced algorithms selected by the system to configure the level of tolerance of the erroneous answer. The algorithms are divided into three categories: Common Abbreviations, Keyboard Fat Fingering (accidentally pressing the nearest neighbor on the keyboard), and Phonetics.

Table 11-4 Answer Logic Algorithm Example

Algorithm	Description	Reason
Common Abbreviations	This algorithm handles common abbreviations, common nicknames, common acronyms, and date format. Looks at file for allowed matches.	If the file contains Mrs=Misses, the match can be made in either direction.
Keyboard Fat Fingering	This algorithm handles answers with typos due to the proximity of keys on a standard keyboard.	"u" is directly to the left of "i" so it is allowed.
Phonetics	This algorithm handles answers that "sound like" the registered answer, regional spelling differences, and common misspellings.	Smiith sounds like Smith.

You can enable or disable the Answer Logic algorithms. You can also configure the strength of some algorithms, such as Keyboard Fat Fingering and Phonetics for evaluating answers given for challenge questions.

This section contains the following topics:

- Understanding Common Response Errors
- Configuring the Levels of Answer Logic

11.10.3.1 Understanding Common Response Errors

This section highlights the most common response errors and shows how Answer Logic algorithms are used for the system to intelligently detect the correct answers in the challenge response process.

Examples of abbreviations, phonetics, and keyboard fat fingering are also provided.

This section contains the following topics:

- About Abbreviations
- About Phonetics
- About Keyboard Fat Fingering

11.10.3.1.1 About Abbreviations

This algorithm handles common abbreviations, common nicknames, common acronyms, and date format.

Common Abbreviations

The algorithm matches the words in the following pairs as equivalent. OAA Admin has predefined list of word-pairs that cover common abbreviations, common nicknames, and common acronyms.

- Street - St.
- Drive - Dr.
- California - CA

Common Nicknames

Oracle has a predefined list of the most common nicknames that is used in the challenge response process. For example:

- Timothy - Tim
- Matthew - Matt

Date Format

The questions that require date as the answer specify the format in which the user should enter the answer. The format is either YYYY or MMDD, but not both. However, from experience, users still use other formats during the challenge response process. The abbreviation logic for date format sees the following as the same:

- 0713
- 713
- July 13th
- July 13
- July 13, 1970

11.10.3.1.2 About Phonetics

This algorithm handles answers that "sound like" the registered answer, regional spelling differences, and common misspellings.

The phonetics algorithm is only supported in English.

Common Misspellings

Oracle's Phonetic Answer Logic algorithm accounts for misspellings.

- ph - f
- Correct word: elephant - Spelling mistake: elefant

11.10.3.1.3 About Keyboard Fat Fingering

This algorithm accounts for typos due to the proximity of keys on a standard keyboard and transposed letters. Answers with typos due to the proximity of keys on a standard keyboard are handled by this algorithm.

The number of fat fingering characters allowed depends on the length of the original word and the level set. The algorithm returns a percentage score associated with the characters that have an exact match. The intensity determines the minimum score required to match the answer with the registered answer.

The fat fingering algorithm is only supported in English.

Common Typos

- Switching "w" and "e"
- Switching "u" and "i"
- Switching "t" and "r"

Examples of Fat Fingering

Correct word: signature - Fat finger: signatire

11.10.3.2 Configuring the Levels of Answer Logic

The level of Answer Logic, the intensity or strength of algorithms, used to evaluate answers given for challenge questions is adjustable.

You can enable or disable each algorithm and you can also specify the following levels for the algorithms used:

- **Off:** No Answer Logic is used. Answers must exactly match those provided at the time of registration.
- **Low:** Low level of Answer Logic is used. Answers provided by the user must be a match or near-match to the answers that were provided at the time of registration.
- **Medium:** More Answer Logic is used. You are given some freedom for the answers that are provided. For instance, St. is acceptable for Street.
- **High:** Highest level of Answer Logic is used. The constraints are not strict for matching.

Note:

The lower the setting the higher the degree of exactness required for acceptance of answers.

For example, high risk transactions such as wire transfers may require a high degree of certainty (i.e. exact match) whereas accessing personal, non-sensitive information may require a lower degree of response certainty. The following example demonstrates how the answer logic algorithm works:

Question: Who was your favorite teacher in high school?

Registered answer: Mrs. Smith

Given answer: Misses Smuth

Logic level: If set to High, the answer is accepted.

Each algorithm generates a score that represents how close the given answer is to the registered answer. You can configure OAA Admin to accept different threshold score ranges for each algorithm individually. Separate threshold values for each algorithm (low/medium/high) are set in a properties file. The default thresholds are described as follows.

Abbreviation

The values are as follows:

- Return values: 0 or 100 (no-match OR match)
- Levels: **On** or **Off**
- Logic:
  - If an abbreviation entry exists linking the given strings, score is 100.
  - Else score is 0.

Fat Fingering
The values are as follows:

- Return values: range 0 to 100
- Levels: **Off**, **Low** (90+), **Medium** (75+), **High** (60+)
- Logic:
  - If the string lengths do not match, score is 0.
  - If a position does not have the expected character or its neighbor, score is 0.
  - Else compute the number of positions that have the neighboring characters.
  - Score = (StringLength – NeighborPositionCount) * 100 /StringLength.

Phonetics
The values are as follows:

- Return values: 0, 60, 75, 90
- Levels: **Off**, **Low** (90), **Medium** (75), **High** (60)
- Logic:
  - Compute primary and alternative phonetic keys for the given strings, using DoubleMetaphone algorithm.
  - If primary keys of both strings match, score is **High**.
  - Else if a primary key of one of the strings and alternate key of the other string match, score is **Medium**.
  - Else if the alternate keys of both string match, score is **Low**.
  - Else the score is **0**.

11.10.4 About Top Categories

The questions are grouped into several categories and the user can select questions from these categories. The Top Categories panel lists the top five categories based on the number of questions linked with a category in descending order.

The standard categories that questions can be grouped into are listed as follows:

- Childhood
- Sports
- Your Birth
- Parents, Grandparents, Siblings
- Children
- Your Employment
- Significant Other
- Pets
- Automobile
- Education
- Miscellaneous

The KBA functionality enables you to manage categories. It allows you to create, edit, delete, and view the categories. If the standard categories that questions can be grouped into do not meet your requirements, then you can create categories that can hold the relevant questions you plan to create. See Create New Category.

11.10.5 About Top Questions

The Top Questions panel lists the five most used challenge questions based on user and validation statistics.

The customer can configure a set of challenge questions that are used to authenticate users. During registration, users are presented with several question menus based on the settings configured in the Registration Logic. For example, the user 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."

The KBA functionality enables you to manage challenge questions. It allows you to create, edit, delete, view, and export and import the challenge questions. If the standard challenge questions do not meet your requirements, then you can create questions as needed. You can also add a validation to the system, if needed. Validations are used to validate the answers given by a user at the time of registration. See Create New Question.

11.10.6 About Disabling Question and Category Logic

The KBA functionality enables you to disable questions and categories.

This section describes the logic to handle disabled questions and categories.

Disabling Logic

The disabling logic is as follows for KBA:

- If you disable the last remaining question in a category, the category is automatically disabled as well.
- The number of active categories must be equal to or greater than the maximum number of categories in the question menu. An error message results when you try to disable a category and this requirement is not met.

Consequences

The following table summarizes the disable results.

Disable Question or Category	New Customers	User with Disabled Question in their Question Set	Users with Question Registered
Question	The disabled question is not used to generate new users' question sets.	At re-registration or when a user changes his preference: Disabled question are replaced with another question from the same category.	The disabled question continues to be active. If the user is re-registering or changing user preference, the disabled question is replaced with another question from the same category.
Category	The disabled category is not used to generate new users' question sets.	At re-registration or when a user changes his preference: All questions in the disabled category are replaced with questions from a new category that has not been used to generate current question set.	Questions from the disabled category continue to be active. If the user is re-registering or changing user preference, all questions in the disabled category are replaced with questions from a new category that has not been used to generate the current question set.

11.10.7 About Deleting Question and Category Logic

The KBA functionality enables you to delete questions and categories.

This section describes the logic to handle deleted questions and categories.

Delete Logic

The logic to delete is as follows for KBA:

- You cannot delete a question that is in use by a registered user.
- Deleted questions are not available for new registrations but the user currently registered for these questions can continue to use them.
- You can delete a category if it is not referenced by questions in use.

11.10.8 Configuring Validations for Answer Registration

You can configure validations that you can use to control the answers a user is allowed to register for all questions at the time of registration.

Validations are used to validate the answers given by a user at the time of registration. For answers, you can restrict the users to alphanumeric and a few specific special characters by adding a Regular Expression validation. For example, if the question, "What year did you start junior high school," is assigned the Month-Day-Year (MMDDYY) validation, a user registering for this question is not allowed to provide "April 1st 1920" for the answer.

You can assign unique validations to each question to control the answers a user is allowed to register. While creating or editing questions, you can assign a validation, by selecting validation type from the **Registration Validation** list. You can also import and export validations.

To learn about validation types, see Create New Validation.

11.11 Configuring Push Notification for Oracle Mobile Authenticator

OAA now allows you to configure push notification for the Oracle Mobile Authenticator (OMA) application.

Topics

- Configuring Oracle Mobile Authenticator Push Notification for Android
- Configuring Oracle Mobile Authenticator Push Notification for iOS

11.11.1 Configuring Oracle Mobile Authenticator Push Notification for Android

The OMA is a mobile device app that uses Time-Based One-Time Password (TOTP) or push notifications to authenticate users with a two-factor authentication scheme. OAA now allows you to configure push notification for the OMA app.

When you are asked to authenticate using a push notification for OMA, then a push notification is delivered to an Android device where you have to either allow or deny the login attempt. The push notification is delivered to the OMA app, which then communicates with the OAA server to grant or deny you access to the protected resource.

Topics

The following topics describe how to configure push notification on Android:

- Installing the Oracle Mobile Authenticator App as an Authentication Method
- Creating a Google Firebase Project Enabled for Google Cloud Messaging
- Configuring OAA Properties for Android Push Notification
- Registering the User Account with Oracle Mobile Authenticator for Android

- Accessing a Protected Application Using Android Push Notification

11.11.1.1 Installing the Oracle Mobile Authenticator App as an Authentication Method

You can download the OMA app for Android devices from the Google Play Store.

Ensure that you have installed the OMA app on your Android device before proceeding.

11.11.1.2 Creating a Google Firebase Project Enabled for Google Cloud Messaging

To send push notifications to Android devices, you must ensure a project is enabled with an Android push notification service. The push notification service that you can use for Android is Google Cloud Messaging (GCM), which requires you to create a Google Firebase project.

Perform the following steps to create a Google Firebase project:

1. Login to Google Firebase console at https://console.firebase.google.com/.
2. Click **Create a project**.
3. In the **Project name** field, enter a name for your project. For instance, OAAAndroidPUSH.
4. Select **I accept the Firebase terms** check box.
5. On the Google Analytics for your Firebase project page, deselect **Enable Google Analytics for this project**, and then click **Create project**.
6. Click **Continue** when your new project is ready.
7. In the left navigation pane of the project window, click the **Settings** icon, and then select **Project settings**.
8. On the **Project settings** page, click **Cloud Messaging**.
9. Note the values present in the **Server key** and **Sender ID** fields. These values are required in the next section for setting up OAA properties.

11.11.1.3 Configuring OAA Properties for Android Push Notification

You must set up some OAA properties that are required for configuring push notifications for Android devices.

The following table lists the OAA properties that you can configure for push notification for Android.

Table 11-5 OAA Properties

Property Name	Description	Sample Value
bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.proxyProtocol	The protocol of the proxy server.	http or https
bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.proxyHost	The host name or IP address of the proxy server.	proxy.example.com

Table 11-5 (Cont.) OAA Properties

Property Name	Description	Sample Value
bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.proxyPort	The port of the proxy server.	80
bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.google.firebase.serverKey	The Firebase "Server Key."	AAAAh1hlXa8:APA91....
bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.google.firebase.senderId	The Firebase "Sender ID."	58213467743

Note:

The `proxyProtocol`, `proxyHost`, and `proxyPort` properties are only required if internet access is available through a proxy server. If OAA has direct access to the internet these properties do not need to be set.

You can configure the OAA properties using the following REST API:

```
PUT  <PolicyUrl>/policy/config/property/v1
```

Note:

In this case remove `/oaa-policy` from the `<PolicyUrl>`, for example use `https://<host>:<port>/policy/config/property/v1` not `https://<host>:<port>/oaa-policy/policy/config/property/v1`

Consider the following example of configuring an OAA property using the CURL command. The example below assumes OAA accesses the internet through a proxy server:

```
curl --location -g --request PUT 'https://<PolicyUrl>/policy/config/
property/v1' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic <Base64Encoded(<username>:<password>)>'
\
--data '[
{
"name":
"bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.proxyProtocol
",
"value": "https"
},
{
"name":
"bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.proxyHost",
"value": "proxy.example.com"
},
{
"name": "bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.proxyPort",
"value": "80"
},
{
"name":
"bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.google.firebase.ser
verKey",
"value":
"AAAAh1hlXa8:APA91bGOGR4pMYe9GC6a2rU169hTCBVmc.............................
.LpU2F8_Egn7IZguC1Rr2HSNnROzXu1d1Lam0TJ"
},
{
"name":
"bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.google.firebase.sen
derId",
"value": "58213467743"
}
]'
```

For details about finding the `PolicyUrl` and authenticating, see OAA Admin API.

For details about the REST API, see Configuration Properties REST Endpoints.

11.11.1.4 Registering the User Account with Oracle Mobile Authenticator for Android

This section provides information about how to register the user account within the OMA application.

Perform the following steps:

1. Log in to the User Preferences console at `https://<SpuiUrl>`.
2. Under **Authentication Factors**, select **Add Authentication Factor**, and then **OMA Push Notification Challenge**.

   The **Add Mobile Device** screen appears.
3. Open the signed OMA app on the Android device.
4. Click **Add Account +**.

   This will launch the camera on your Android device.
5. Use the camera to scan the QR code on the screen.

   The **Login Required** screen appears.
6. Do the following:
   a. In the **Username** field, enter the user name displayed on the **User Preferences console** screen as the user name is case sensitive.
   b. In the **PIN code** field, enter the PIN code displayed on the **User Preferences console** screen.
7. Click **Sign in** and accept the certificate if prompted.

   The account is successfully added in OMA.

8. On the **User Preferences console** screen, click **Done**.

The OMA Push Notification Challenge for the registered device appears in the User Preferences console.

11.11.1.5 Accessing a Protected Application Using Android Push Notification

To test the push notification you must access a protected application.

Perform the following steps to access a protected application:

1. Access the protected application. For example, `https://www.example.com/application`.

   The **OAA challenge choice** screen appears.

2. Under **OMA Push Notification Challenge**, select **Approve login on device <DeviceID>**.

   You are redirected to the PUSH screen where a notification should appear on your Android device.

3. Select **Allow** on the device to login.

If authentication is successful, you are redirected to the protected page.

11.11.2 Configuring Oracle Mobile Authenticator Push Notification for iOS

OAA now allows you to configure push notification for the OMA app for iOS.

When you are asked to authenticate using a push notification for OMA, then a push notification is delivered to an iOS device where you have to either allow or deny the login attempt. The push notification is delivered to the OMA app, which then communicates with the OAA server to grant or deny you access to the protected resource.

Push notifications are sent to the iOS device through Apple's Push Notification service (APNS). This requires an Apple Push notification certificate, which is only generated from the Apple Developer's Console.

The standard OMA application installed directly from the Apple App Store do not support push notifications for OAA login attempts. The push notification certificate generated from the Apple Developer Console is tied directly to the OMA application. Therefore, a custom OMA application must be built and signed by the same certificate to receive push notifications.

When you register the iOS device with OAA, a device ID is stored for the user (visible from User Preferences console) and this is used to identify the desired recipient.

Apple push notification certificates are built/signed by Apple specifically for their production or development servers. A development certificate cannot be used to send push notifications to the production APNS server and vice-versa. If using an APNS production certificate, you must request this from Apple and use it in the APNSCertificate.jks. This certificate is then used to sign the custom built OMA application. Likewise, if you are using an APNS development certificate, then you must request this from Apple and use it in the APNsCertificate.jks, which is then used to sign the custom built OMA application.

Topics

The following topics describe how to configure push notification on iOS:

11.11.2.1 Creating an Apple iOS Certificate, App ID, Bundle Identifier, and Keystore

Learn to create an Apple iOS Certificate, App ID, Bundle Identifier, and Keystore.

See document ID 2319759.1 in My Oracle Support for instructions to create an Apple iOS Certificate, App ID, Bundle Identifier, and Keystore.
After completing the steps mentioned in document ID 2319759.1, you must return to this documentation for further instructions.

11.11.2.2 Copying the APNS Java Key Store to OAA

After creating the `APNSCertificate.jks` file, you must copy this file to the `<NFS_VAULT_PATH>` which maps to `/u01/oracle/service/store/oaa`.

To copy the file to a file based vault, perform the following steps:

1. Create a directory in the NFS volume `<NFS_VAULT_PATH>`:

```
$ cd <NFS_VAULT_PATH>
$ mkdir -p ChallengeOMAPUSH/apns
$ cp APNSCertificate.jks <NFS_PATH>/ChallengeOMAPUSH/apns
$ sudo chmod 444 <NFS_VAULT_PATH>/ChallengeOMAPUSH/apns/
APNSCertificate.jks
```

Note:

- You can copy the `APNSCertificate.jks` to any location inside the `<NFS_VAULT_PATH>`, however you must change the property `bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.apns.keystorePath` to point to the directory where the file is copied to. See Configuring OAA Properties for iOS Push Notification.

11.11.2.3 Configuring OAA Properties for iOS Push Notification

You must set up some OAA properties that are required for configuring push notification for iOS devices.

The following table lists the OAA properties that you can configure for push notification for iOS.

Table 11-6 OAA Properties

Property Name	Description	Sample Value
bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.proxyProtocol	The protocol of the proxy server.	http or https
bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.proxyHost	The host name or IP address of the proxy server.	proxy.example.com
bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.proxyPort	The port of the proxy server.	80
bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.apns.keystorePath	The location of the APNSCertificate.jks keystore.	/u01/oracle/service/store/oaa/ChallengeOMAPUSH/apns/APNSCertificate.jks
bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.apns.keystorePass	The keystore password.	<password>
bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.apns.h2Topic	The APNS App ID created on the Apple Developer console.	com.example.MyApp

Note:

The `proxyProtocol`, `proxyHost`, and `proxyPort` properties are only required if internet access is available through a proxy server. If OAA has direct access to the internet these properties do not need to be set

You can configure the OAA properties using the following REST API:

```
PUT  <PolicyUrl>/policy/config/property/v1
```

Note:

In this case remove `/oaa-policy` from the `<PolicyUrl>`, for example use `https://<host>:<port>/policy/config/property/v1` not `https://<host>:<port>/oaa-policy/policy/config/property/v1`.

Consider the following example of configuring an OAA property using the CURL command. The example below assumes OAA accesses the internet through a proxy server:

```
curl --location -g --request PUT 'https://<PolicyUrl>/policy/config/
property/v1' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic <Base64Encoded(<username>:<password>)>' \
--data '[
{
"name":
"bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.proxyProtocol",
"value": "https"
},
{
"name": "bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.proxyHost",
"value": "proxy.example.com"
},
{
"name": "bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.proxyPort",
"value": "80"
},
{
"name":
"bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.apns.keystorePath",
"value": "/u01/oracle/service/store/oaa/ChallengeOMAPUSH/apns/
APNsCertificate.jks"
},
{
"name":
"bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.apns.keystorePass",
"value": "<password>"
},
{"name":
"bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.apns.h2Topic",
"value": "com.example.MyApp"}
]'
```

For details about the REST API, see Configuration Properties REST Endpoints.

11.11.2.4 Registering the User Account with Oracle Mobile Authenticator for iOS

This section provides information about how to register the user account within the OMA application.

Perform the following steps:

1. Log in to the User Preferences console at `https://<SpuiUrl>`.
2. Under **Authentication Factors**, select **Add Authentication Factor**, and then **OMA Push Notification Challenge**.

   The **Add Mobile Device** screen appears.
3. Open the signed OMA app on the iOS device.
4. Click **Add Account +**.

This will launch the camera on your iOS device.

5. Use the camera to scan the QR code on the screen.

   The **Login Required** screen appears.

6. Do the following:

   a. In the **Username** field, enter the user name displayed on the **User Preferences console** screen as the user name is case sensitive.

   b. In the **PIN code** field, enter the PIN code displayed on the **User Preferences console** screen.

7. Click **Sign in** and accept the certificate if prompted.

   The account is successfully added in OMA.

8. On the **User Preferences console** screen, click **Done**.

The OMA Push Notification Challenge for the registered device appears in the User Preferences console.

11.11.2.5 Installing the Oracle Mobile Authenticator

The standard OMA application installed directly from the Apple App Store does not support push notifications for OAA login attempts.

The push notification certificate generated from the Apple Developer Console is tied directly to the OMA application. Therefore, you must build a custom OMA application and get it signed by the same certificate to receive push notifications.

See document ID 2319759.1 in My Oracle Support for instructions on how to create this custom OMA application.

11.11.2.6 Accessing a Protected Application Using iOS Push Notification

To test the push notification you must access a protected application.

Perform the following steps to access a protected application:

1. Access the protected application. For example, `https://www.example.com/application`.

   The **OAA challenge choice** screen appears.

2. Under **OMA Push Notification Challenge**, select **Approve login on device <DeviceID>**.

   You are redirected to the PUSH screen where a notification should appear on your iOS device.

3. Select **Allow** on the device to login.

If authentication is successful, you are redirected to the protected page.

12 Integrating OAA with Other Products

OAA allows integration with other products to support Multi-factor Authentication (MFA) either through REST APIs or browser-based flows.

OAA can be integrated with clients supporting browser-based user flows, for example Oracle Access Management (OAM) and Oracle Identity Manager (OIM), or REST API based user flows such as Oracle RADIUS Agent (ORA) or custom developed applications using REST API's.

12.1 Integrating OAA with OAM

OAA can be integrated with OAM using the `OAAAuthnPlugin` and registering OAA as a TAP partner.

OAA Interaction with OAM to Provide Multi Factor Authentication

The following provides an overview of the user interaction flow for OAA-OAM integration through a browser-based flow.

1. The user accesses the OAM (Webgate) protected resource through the browser.
2. The user is redirected to OAM for authentication.
3. OAM presents the Login Screen to the user and after authentication redirects the flow to OAA with the TAP Token for multi-factor authentication.

   Note:

   OAA integrates with OAM using the **OAAAuthnPlugin** and by registering OAA as a TAP partner.

4. OAA presents the user with the additional challenge pages with factors for authentication.
5. After the challenge flow is complete, user is redirected back to OAM with success or failure messages.
6. User is granted access to the resource if the multi-factor authentication was successful.

Configuring OAM with OAA

To configure OAM with OAA, see the tutorial Integrate Oracle Access Management with Oracle Advanced Authentication .

12.2 Integrating OAA with ORA

OAA can be integrated with Oracle RADIUS Agent (ORA) using REST APIs.

OAA Interaction with ORA to Provide Multi Factor Authentication

In the example below an Oracle Database is integrated with ORA and OAA.

1. User logs in into the Database with a database client (sqlplus) and the user credentials (username/password) are verified.
2. After authentication, the database invokes ORA for the second factor authentication.
3. ORA invokes an API to determine the user challenge and presents a challenge prompt for the user:
    a. OAA provides the challenge prompt information.
    b. User is shown a prompt and is asked for an answer by ORA.
4. ORA redirects to OAA and it validates the answer provided by ORA.
5. ORA redirects back for resuming the database login session.
6. User is granted access to the database if the challenge validation was successful.

Configuring ORA with OAA

To configure ORA with OAA, see the tutorial Use Oracle RADIUS Agent with Oracle Advanced Authentication for Multi-Factor Authentication.

12.3 Integrating OAA with OIM

You can implement the password management feature for OAA-protected applications by integrating OAA with Oracle Identity Manager (OIM).

The **Forgot Password** feature allows the user to request a password reset. Oracle Identity Management (OIM) exposes REST APIs for password management. OAA uses these REST APIs to reset the user password.

The following topics provides an overview of the user interaction flow for OAA-OIM integration through OIM REST APIs.

Topics

- Understanding the Forgot Password Flow for OAA and OIM Integration
- Configuring the Forgot Password Feature

12.3.1 Understanding the Forgot Password Flow for OAA and OIM Integration

The Forgot Password flow allows the users to reset their password after successfully answering all challenge questions.

Note:

You must ensure that OIM is integrated with OAM prior to following these steps.

Consider a scenario where the user is at the OAM Login page and clicks the "Forgot Password" link. The Forgot Password feature is implemented as part of OAA. OAM redirects the user to the OAA "Forgot Password" URL, and passes the destination URL to which OAA must redirect upon a successful password change as a query parameter (backURL).

The flow of interactions between the components is as follows:

1. A user tries to access a resource protected by OAM.
2. The OAM Webgate (SSO Agent) intercepts the request and redirects the user to the Oracle Access Manager Login Page.
3. The user clicks on the **Forgot Password** link on the Oracle Access Manager Login page, which sends the user to the OAA Forgot Password URL.
4. The user enters the username, which is redirected to KBA for authentication to proceed further to reset the password.
5. OAA interacts with the user to enable the user to reset the password.
6. Navigate to the application URL to which access was attempted in step 1. Specify your credentials to log in.

12.3.2 Configuring the Forgot Password Feature

You can integrate OAA with OIM using the OIM REST APIs to configure the forgot password feature.

Prerequisite: You must ensure that OIM is integrated with OAM prior to following these steps.

There are three configurations that you must perform to implement the Forgot Password feature:

1. Configuring the OAA Admin Console. See Configuring OAA for OIM Integration.
2. Establishing a connection between OAA and OIM. See Configuring OIM Properties for Integration.
3. Configuring OAM and OIM integration. See Configuring OAM Forgot Password Link.

12.3.2.1 Configuring OAA for OIM Integration

You must create an integration agent to integrate client applications with OAA. Then, you must create assurance levels for an integration agent and define rules for an assurance level. You can perform these tasks using the OAA Administration UI console.

Perform the following steps in the OAA Administration UI console:

1. Login to the OAA Administration console `https://<AdminUrl>`. You are redirected to the OAM login page, as the console is protected by OAM OAuth. Specify your credentials and login.
2. Under **Quick Actions** select **Create Other Integration Agent**.

3. Click **Manage Integration Agents**.
4. In the **Create Integration Agent** window, specify the following:
   a. **Name**: Enter a name for the integration agent, for instance, **ForgotPasswordFlowAgent**.
   b. **Description**: Add a description about the integration agent.
   c. **Integration Agent Type**: From the list, select **API**.
   d. Click **Save**.
5. In the **Integration Agents** window, click the integration agent for which you need to create the assurance level, for instance **ForgotPasswordFlowAgent** link.
6. Under the **Assurance Levels** tab, click **Create**.
7. Specify the required details:
   a. **Name:** Specify the name for this assurance level, for instance, **ForgotPasswordAL**.
   b. **Description:** Provide the description for the assurance level.
8. Click **Create**.
9. Under the **Assurance Levels** tab, click the required assurance level for which you are required to define rules, for instance **ForgotPasswordAL** link.
10. Under **Uses** select the required factors to assign to the assurance level. In this scenario, select **Security Question Challenge** *only*.
11. Click **Save**.

12.3.2.2 Configuring OIM Properties for Integration

OAA uses the REST APIs to communicate with OIM for all the user operations. Therefore, you need to establish a connection between OAA and OIM so that the integration happens seamlessly.

Use the `<PolicyUrl>/policy/config/property/v1` REST API to configure properties.

Note:

In this case remove `/oaa-policy` from the `<PolicyUrl>`, for example use `https://<host>:<port>/policy/config/property/v1` not `https://<host>:<port>/oaa-policy/policy/config/property/v1`

For details about finding the `PolicyUrl` and authenticating, see OAA Admin API.

For details about the Configuration Properties REST Endpoint, see Configuration Properties REST Endpoints

You need to set the following OIM properties as per your environment, for example:

```
curl --location -g --request PUT 'https://<PolicyUrl>/policy/config/
property/v1' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic <Base64Encoded(<username>:<password>)>'
```

```
\
--data '[
{
"name": "oaa.default.user.management.provider.enum.oim.url",
"value": "https://<OIM Managed Server>:<OIM Managed Port>"
},
{
"name": "oaa.default.user.management.provider.enum.oim.admin.username",
"value": "<Username of Oracle Identity Manager Administrator>"
},
{
"name": "oaa.default.user.management.provider.enum.oim.admin.password",
"value": "<Password of Oracle Identity Manager Administrator>"
},
{
"name": "oaa.default.user.management.provider.enum.oim.oaaDefaultGroup",
"value": "<Default Group Name of User>"
}
]'
```

Note:

- The <*username*> and <*password*> in the preceding command should be related to oaa-policy as they are policy and oaa-related users. Here, <*username*> refers to <RELEASENAME>-oaa, and <*password*> refers to <Base64Decoded(oaaapikey)>. For instance, `idmenv0025-oaa-policy` could be used as a user name.
- You must note that when you make a PUT call from `https://<host>:port>/policy/config/property/v1` and then a GET call from `https://<host>:port>/oaa/runtime/config/property/v1?propertyName=oaa.default.user.management.provider.enum.oim`, this property requires a few seconds (typically 30 seconds) to retrieve the details from the OAA service config API.

There are some optional OIM properties that you can configure using the REST APIs.

Table 12-1 Optional OIM Properties

Property	Description
`oaa.default.user.management.provider.enum.oim.forgotPassword.queryParamNameForSuccessRedirectUrl=backUrl`	The value of the parameter is the query parameter name. It refers to the value that you provide for the redirect URL where the user is redirected after successful password change. **Note:** You must ensure that the value of `backUrl` begins with `http:// or https://` as follows: `https://<host>:<port>/oaa/usermgmt?ojr=forgotpasswordusername&backUrl=https://example.com/`

Table 12-1 (Cont.) Optional OIM Properties

Property	Description
`oaa.default.user.management.provider.enum.oim.forgotPassword.successRedirectUrl`	It refers to the redirect URL to which the user is redirected after successful password change. **Note:** If the `successRedirectUrl` property is present along with `queryParamNameForSuccessRedirectUrl` property, then the `successRedirectUrl` takes precedence.
`oaa.default.user.management.provider.enum.oim.forgotPassword.ui.configErrorMessage`	It refers to the error that is thrown if there is an OIM configuration issue. It is as follows: "`There is an error, please check with your system administrator.`"
`oaa.default.user.management.provider.enum.oim.forgotPassword.ui.heading`	It refers to the heading that you provide for the Forgot password screen.
`oaa.default.user.management.provider.enum.oim.forgotPassword.ui.userNotFoundMessage`	It refers to the error message, which is thrown if the user account is not found.
`oaa.default.user.management.provider.enum.oim.forgotPassword.ui.factorNotConfigured`	It refers to challenge questions not configured.

12.3.2.3 Configuring OAM Forgot Password Link

You must configure the OAM and OIM integrated environment so that the Forgot Password link points to OAA instead of OIM for password reset.

Use the following CURL command to update OAM to point to OAA for resetting the password:

```
curl --user weblogic_idm:<password> -i -H "Content-Type:application/
json" -H "Accept: */*" \
-X PUT -d '{"forgotPasswordURL":"https://<SpuiUrl>/oaa/usermgmt"}' \
http://<OAM host>:<OAM port>/oam/admin/api/v1/configurationService/
forgotPassword
```

Note:

For detail on how to find the `<SpuiUrl>`, see Printing Deployment Details

12.4 Integrating OAA with other Applications

OAA can be intergrated with other applications using REST API's.

The Oracle Advanced Authentication REST APIs provide a way to integrate Oracle Advanced Authentication with REST clients so developers can create applications that can use Oracle Advanced Authentication. For example, application developers can use REST API's to create applications for OAA administration, allow end users to set their factor preferences, or challenge and validate end users for second factor authentication with OAA.

For examples of common REST API calls, see Use Oracle Advanced Authentication REST APIs with Postman.

For a full list of REST API's, see:

- OAA Admin API
- OAA Runtime API

13 Customizing OAA

13.1 Customizing Email and SMS Messaging Provider

You can customize email and SMS message provider by implementing the `MessagingProvider` interface.

Prerequisites

Ensure you have the following prerequisites before proceeding:

- Oracle Linux 7.x environment
- JDK 1.8.x for compilation
- Apache Maven 3.6.2 or later
- Any third-party jars necessary for your implementation

Perform the following to customize email and SMS Messaging Provider:

1. Implement Custom Logic and Create Jar files
2. Integrate the Implementation with OAA
3. Make the Implementation Available at Runtime

Implement Custom Logic and Create Jar files

1. Download the project zip file from the following location in the management container: `/u01/oracle/libs/messagingprovider-interface-12.2.1.4.1-<date>.jar` and extract it to your working directory.
2. Create an implementation of your custom logic for email and SMS. For more information about the interface and methods, see the Javadoc reference.

   Note:

   E-mail and SMS must have separate implementation classes.

   In the implementation class that implements the sender interface, make sure that `@Service` declaration is created with necessary imports. This declaration helps the framework to load the custom implementation at runtime.

3. Make changes to the `pom` file for compiling and generating a jar file that includes this implementation. Ensure the following:

- The `jersey-hk2` dependency must be declared as dependency in the `pom` file for the build.
- The `MessagingProvider` interface must be used and declared as dependency in the `pom` file for the build.
- The implementation must generate one jar and additional third-party jars. You can choose to implement the send method using one or more classes and package them into this jar file.

4. Test the implementation to make sure that it works as necessary.

Integrate the Implementation with OAA

1. Configure the custom implementation class in OAA.
   - For e-mail:
     Update the `customProvider` property of the `ChallengeEmail` enum with fully qualified class name of the implementation class.

     For example, if the name of the implementation class is `com.company.MyCustomEmailMessagingSender`, update the `ChallengeEmail` enum property as `bharosa.uio.default.challenge.type.enum.ChallengeEmail.customizedProvider=com.company.MyCustomEmailMessagingSender`

     To do this, use the configuration property REST API as shown in the following sample request:

     ```
     curl  --request PUT 'https://<PolicyUrl>/policy/config/
     property/v1' \
     --header 'Content-Type: application/json' \
     --header 'Authorization: Basic
     <Base64Encoded(<username>:<password>)>' \
     --data '[
     {
     "name":
     "bharosa.uio.default.challenge.type.enum.ChallengeEmail.customize
     dProvider"
     "value": "com.company.MyCustomEmailMessagingSender"
     }
     ]'
     ```

     Note:

     In this case, and elsewhere in this section, remove `/oaa-policy` from the `<PolicyUrl>`, for example use `https://<host>:<port>/policy/config/property/v1` not `https://<host>:<port>/oaa-policy/policy/config/property/v1`

     For details about finding the `PolicyUrl` and authenticating, see OAA Admin API.

     For details about the Configuration Properties REST Endpoint, see Configuration Properties REST Endpoints

- For SMS:
  Update the `customProvider` property of the `ChallengeSMS` enum with fully qualified class name of the implementation class.

  For example, if the name of the implementation class is `com.company.MyCustomSMSMessagingSender`, update the `ChallengeSMS` enum property as `bharosa.uio.default.challenge.type.enum.ChallengeSMS.customizedProvider=com.company.MyCustomSMSMessagingSender`

  To do this, use the configuration property REST API as shown in the following sample request:

```
curl  --request PUT 'http://<PolicyUrl>/policy/config/property/v1' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic
<Base64Encoded(<username>:<password>)>' \
--data '[
{
"name":
"bharosa.uio.default.challenge.type.enum.ChallengeSMS.customizedProvid
er"
"value": "com.company.MyCustomSMSMessagingSender"
}
]'
```

Make the Implementation Available at Runtime

When the e-mail or SMS service pods are configured/started, shared volume information is made part of that configuration. It appears in the `deployment.yaml` file of that chart.

Create a persistent NFS volume and provide that information in the `values.yaml` for the email chart as shown in the following sample.

Note:

Do not change the `mountPathPrefix` value

```
# volume to store customized email sending implementation
customizedFactorImplVolume:
  # name of the volume
  name: "nfsvolume"
  # server where the volume is located
  server: <NFS_IP_ADDRESS>
  # path on the server where the volume is located
  path: <NFS_PATH>/FactorProviderImpls
  # prefix of volume's mounted path in email container
  mountPathPrefix: /u01/oracle/
  # relative path of mounted volume, relative to the above prefix
  mountRelativePath: <NFS_VOLUME>/customprovider
  # indicate whether the volume should be readOnly
  readOnly: false
  # names of customizedJars expected in mounted volume, separated by
comma
  customizedJars: "OAACustomMessaging-Provider.jar"
```

If the external volume is not NFS then perform the following:

1. Edit the `deployment.yaml` file in the email chart as shown:

   ```
   {{- if .Values.customizedFactorImplVolume }}
           - name: {{.Values.customizedFactorImplVolume.name}}
             nfs:
               server: {{.Values.customizedFactorImplVolume.server}}
               path: {{.Values.customizedFactorImplVolume.path}}
   {{- end }}
   ```

2. Copy the custom implementation and third party dependency jar files to the following folder: `<mountPathPrefix><mountRelativePath>/`.

Stop and start the nodes in the pod to start loading your custom implementation.

13.2 Customizing the OAA User Interface

You can customize certain features of the OAA user interface (UI), such as the Administration Console UI, User Preferences Console UI, and the Runtime UI using the configuration properties.

You can customize the UI by setting configuration properties using the REST API:

```
PUT  <PolicyUrl>/policy/config/property/v1
```

> **Note:**
>
> In this case remove `/oaa-policy` from the `<PolicyUrl>`, for example use `https://<host>:<port>/policy/config/property/v1` not `https://<host>:<port>/oaa-policy/policy/config/property/v1`.

For details about finding the `PolicyUrl` and authenticating, see OAA Admin API.

For details about the Configuration Properties REST Endpoint, see Configuration Properties REST Endpoints

You must bear the following points in mind while setting the configuration property:

- Image type must be png, jpg, or jpeg.
- Image values must be set to an existing image available on an external URL.
- When updating the footer parameters, ensure that you update all the footer-specific parameters in one go for the changes to be visible in the UI.
- The default out-of-the box values are internal only and are not displayed when fetched. For example, GET `<PolicyUrl>/policy/config/property/v1?propertyName=<property>` returns `[]` as the default value. Only when a property is custom defined, the value is returned with GET.

- To reset a property to the out-of-the-box default value, use `DELETE <PolicyUrl>/policy/config/property/v1?propertyName=<property>`.

The following sections describe how to customize the various UI's by setting their configuration properties.

- Configuration Properties to Customize the Administration Console UI
- Configuration Properties to Customize the User Preferences Console UI
- Configuration Properties to Customize the Runtime UI
- Configuration Values for Generic Font Families

13.2.1 Configuration Properties to Customize the Administration Console UI

Learn about the configuration properties that you can set to customize the Administration Console UI.

Table 13-1 Configuration Properties to Customize the Administration Console UI

Property Name	Description	Sample Value
oaa.rui.ui.theme.default.image.path.logo	Logo image path	https://www.example.com/content/images/logo.jpg
oaa.admin.ui.theme.default.image.path.background	Background image path	https://www.example.com/content/images/background.jpg
oaa.admin.ui.theme.default.image.path.favicon	Favicon image path	https://www.example.com/content/images/favicon.jpg
oaa.admin.ui.theme.default.font.text.header	Application header text	Example Company Advanced Authentication
oaa.admin.ui.theme.default.font.color.header	Application header color	#ffffff
oaa.admin.ui.theme.default.font.text.title	Application title text	Example Company Advanced Authentication
oaa.admin.ui.theme.default.footer.color	Footer text color	#00688c
oaa.admin.ui.theme.default.footer.color.copyrightNotice	"copyrightNotice" text color	rgba(22, 21, 19, .6)
oaa.admin.ui.theme.default.footer.text.about	Footer text for "about"	About Example Company
oaa.admin.ui.theme.default.footer.text.contactus	Footer text for "contactus"	Contact us
oaa.admin.ui.theme.default.footer.text.legalnotice	Footer text for "legalnotice"	Legal Notice
oaa.admin.ui.theme.default.footer.text.termsofuse	Footer text for "termsofuse"	Terms of use
oaa.admin.ui.theme.default.footer.text.privacyright	Footer text for" privacyright"	Privacyright
oaa.admin.ui.theme.default.footer.link.about	Footer "about" link	http://www.example.com/us/corporate/index.html#menu-about
oaa.admin.ui.theme.default.footer.link.contactus	Footer "contactus" link	http://www.example.com/us/corporate/contact/index.html

Table 13-1 (Cont.) Configuration Properties to Customize the Administration Console UI

Property Name	Description	Sample Value
oaa.admin.ui.theme.default.footer.link.legalnotice	Footer "legalnotice" link	http://www.example.com/us/legal/index.html
oaa.admin.ui.theme.default.footer.link.termsofuse	Footer termsofuse link	http://www.example.com/us/legal/terms/index.html
oaa.admin.ui.theme.default.footer.link.privacyright	Footer privacyright link	http://www.example.com/us/legal/privacy/index.html
oaa.admin.ui.theme.default.footer.text.copyrightNotice	Footer text for copyright	Copyright © 2021, Example Company and/or its affiliates. All rights reserved.
oaa.admin.ui.theme.default.image.tiled.background	If specified as true, background image will appear in a tiled manner	FALSE
oaa.admin.ui.theme.default.font.family	If specified, custom font family name will be used. Refer to <Generic font families>	Oracle Sans

13.2.2 Configuration Properties to Customize the User Preferences Console UI

Learn about the configuration properties that you can set to customize the User Preferences Console UI.

Table 13-2 Configuration Properties to Customize the User Preferences Console UI

Property Name	Description	Sample Value
oaa.prefs.ui.theme.default.image.path.logo	Logo image path	https://www.example.com/content/images/logo.jpg
oaa.prefs.ui.theme.default.image.path.background	Background image path	https://www.example.com/content/images/background.jpg
oaa.prefs.ui.theme.default.image.path.favicon	Favicon image path	https://www.example.com/content/images/favicon.jpg
oaa.prefs.ui.theme.default.font.text.header	Application header text	Example Company Advanced Authentication
oaa.prefs.ui.theme.default.font.color.header	Application header color	#ffffff
oaa.prefs.ui.theme.default.font.text.title	Application title text	Example Company Advanced Authentication
oaa.prefs.ui.theme.default.footer.color	Footer text color	#00688c
oaa.prefs.ui.theme.default.footer.color.copyrightNotice	"copyrightNotice" text color	rgba(22, 21, 19, .6)
oaa.prefs.ui.theme.default.footer.text.about	Footer text for "about"	About Example Company

Table 13-2 (Cont.) Configuration Properties to Customize the User Preferences Console UI

Property Name	Description	Sample Value
oaa.prefs.ui.theme.default.footer.text.contactus	Footer text for "contactus"	Contact us
oaa.prefs.ui.theme.default.footer.text.legalnotice	Footer text for "legalnotice"	Legal Notice
oaa.prefs.ui.theme.default.footer.text.termsofuse	Footer text for "termsofuse"	Terms of use
oaa.prefs.ui.theme.default.footer.text.privacyright	Footer text for" privacyright"	Privacyright
oaa.prefs.ui.theme.default.footer.link.about	Footer "about" link	http://www.example.com/us/corporate/index.html#menu-about
oaa.prefs.ui.theme.default.footer.link.contactus	Footer "contactus" link	http://www.example.com/us/corporate/contact/index.html
oaa.prefs.ui.theme.default.footer.link.legalnotice	Footer "legalnotice" link	http://www.example.com/us/legal/index.html
oaa.prefs.ui.theme.default.footer.link.termsofuse	Footer termsofuse link	http://www.example.com/us/legal/terms/index.html
oaa.prefs.ui.theme.default.footer.link.privacyright	Footer privacyright link	http://www.example.com/us/legal/privacy/index.html
oaa.prefs.ui.theme.default.footer.text.copyrightNotice	Footer text for copyright	Copyright © 2021, Example Company and/or its affiliates. All rights reserved.
oaa.prefs.ui.theme.default.image.tiled.background	If specified as true, background image will appear in a tiled manner	false
oaa.prefs.ui.theme.default.font.family	If specified, custom font family name will be used. Refer to <Generic font families>	Oracle Sans
oaa.prefs.ui.theme.default.menu.button.color	To change the color of the menu button	rgb(31,92,255)
oaa.prefs.ui.theme.default.primary.button.color.focus	To change the color of the submit button	rgb(31,92,255)
oaa.prefs.ui.theme.default.primary.button.color.hover	To change the color of the submit button	rgb(31,92,255)
oaa.prefs.ui.theme.default.primary.button.color.active	To change the color of the submit button	rgb(31,92,255)
oaa.prefs.ui.theme.default.header.bar.color	To change the color of the header bar	rgb(99,99,0)
oaa.prefs.ui.theme.default.footer.bar.color	To change the color of the footer bar	rgb(99,99,0)

13.2.3 Configuration Properties to Customize the Runtime UI

Learn about the configuration properties that you can set to customize the Runtime UI.

Table 13-3 Configuration Properties to Customize the Runtime UI

Property Name	Description	Sample Value
oaa.rui.ui.theme.default.image.path.logo	Logo image path	https://www.example.com/content/images/logo.jpg
oaa.rui.ui.theme.default.image.path.background	Background image path	https://www.example.com/content/images/background.jpg
oaa.rui.ui.theme.default.image.path.favicon	Favicon image path	https://www.example.com/content/images/favicon.jpg
oaa.rui.ui.theme.default.font.text.title	Application title text	Example Company Advanced Authentication
oaa.rui.ui.theme.default.footer.color	Footer text color	#00688c
oaa.rui.ui.theme.default.footer.color.copyrightNotice	"copyrightNotice" text color	rgba(22, 21, 19, .6)
oaa.rui.ui.theme.default.footer.text.about	Footer text for "about"	About Example Company
oaa.rui.ui.theme.default.footer.text.contactus	Footer text for "contactus"	Contact us
oaa.rui.ui.theme.default.footer.text.legalnotice	Footer text for "legalnotice"	Legal Notice
oaa.rui.ui.theme.default.footer.text.termsofuse	Footer text for "termsofuse"	Terms of use
oaa.rui.ui.theme.default.footer.text.privacyright	Footer text for" privacyright"	Privacyright
oaa.rui.ui.theme.default.footer.link.about	Footer "about" link	http://www.example.com/us/corporate/index.html#menu-about
oaa.rui.ui.theme.default.footer.link.contactus	Footer "contactus" link	http://www.example.com/us/corporate/contact/index.html
oaa.rui.ui.theme.default.footer.link.legalnotice	Footer "legalnotice" link	http://www.example.com/us/legal/index.html
oaa.rui.ui.theme.default.footer.link.termsofuse	Footer termsofuse link	http://www.example.com/us/legal/terms/index.html
oaa.rui.ui.theme.default.footer.link.privacyright	Footer privacyright link	http://www.example.com/us/legal/privacy/index.html
oaa.rui.ui.theme.default.footer.text.copyrightNotice	Footer text for copyright	Copyright © 2021, Example Company and/or its affiliates. All rights reserved.
oaa.rui.ui.theme.default.image.tiled.background	If specified as true, background image will appear in a tiled manner	false
oaa.rui.ui.theme.default.font.family	If specified, custom font family name will be used. Refer to <Generic font families>	Oracle Sans
oaa.rui.ui.theme.default.button.color.active	Active button color	rgb(79, 105, 63)

Table 13-3 (Cont.) Configuration Properties to Customize the Runtime UI

Property Name	Description	Sample Value
oaa.rui.ui.theme.default.button.color.hover	Hovered button color	rgb(87, 115, 70)
oaa.rui.ui.theme.default.button.color.focus	Focused button color	rgb(95, 125, 79)
oaa.rui.ui.theme.default.font.color.factor	Text color	rgb(22, 21, 19)
oaa.rui.ui.theme.default.font.color.factorlink	Link color	#00688c
oaa.rui.ui.theme.default.font.color.label	Label color	rgba(22, 21, 19, .6)
oaa.rui.ui.theme.default.font.color.header	Factor header color	rgb(22, 21, 19)

13.2.4 Configuration Values for Generic Font Families

Learn about the possible configuration values that you can set for the generic font families.

Table 13-4 Configuration Values for Generic Font Families

Font family	Possible Values
'sans-serif': normal fonts without serifs	Arial Helvetica Verdana Trebuchet MS Gill Sans Noto Sans Avantgarde TeX Gyre Adventor URW Gothic L Optima Arial Narrow
'serif': normal fonts with serifs	Times Times New Roman Didot Georgia Palatino URW Palladio L Bookman URW Bookman L New Century Schoolbook TeX Gyre Schola American Typewriter

Table 13-4 (Cont.) Configuration Values for Generic Font Families

Font family	Possible Values
'monospace': fixed-width fonts	Andale Mono Courier New Courier FreeMono OCR A Std DejaVu Sans Mono
'cursive': fonts that emulate handwriting	Comic Sans MS Comic Sans Apple Chancery Bradley Hand Brush Script MT Brush Script Std Snell Roundhand URW Chancery L
'fantasy': decorative fonts, for titles, etc.	Impact Luminari Chalkduster Jazz LET Blippo Stencil Std Marker Felt Trattatello

14 Understanding Partitioned Schemas

From OAA122141-20221019 onwards, OAA uses a partitioned schema to allow for maintenance of transaction data. Each month, a large amount of data is entered into the transaction tables. As a result, it is necessary to clean up old data entries periodically. Administrators can also purge and archive data to release data that is no longer required

Topics:

- Partition Maintenance
- Viewing Scheduled Jobs and Logs
- Archiving and Purging

14.1 Partition Maintenance

Partition Maintenance is performed using three database stored procedures.

The procedures used for partition maintenance are as follows:

- `SP_OAA_ADD_MONTHLY_PARTITION`
- `SP_OAA_ADD_WEEKLY_PARTITION`
- `SP_OAA_DROP_PARTITION`

The DBMS_SCHEDULER package runs preconfigured jobs, which in turn execute the procedures `SP_OAA_ADD_MONTHLY_PARTITION` and `SP_OAA_ADD_WEEKLY_PARTITION` against the relevant tables in order to create partitions for new data entries.

The scheduler runs both of these procedures periodically to add table partitions. Each table partition will store data whose creation_time are lower than the high value of the table partition. The high value of the table partition is the maximum value allowed for the creation_time column in one data entry.

The `SP_OAA_ADD_MONTHLY_PARTITION` stored procedure adds partitions for tables with a monthly frequency. The script runs at the end of each month to create partitions for the following month.

The `SP_OAA_ADD_WEEKLY_PARTITION` stored procedure adds partitions for tables with a weekly frequency. The script runs at the end of each week to create partitions for the following week.

The `SP_OAA_DROP_PARTITION` procedure is run manually by the Adminstrator to drop the table partitions for tables whose high value is smaller than current_date-retention_days. This procedure is usually run after old data is purged and archived. See Archiving and Purging.

14.2 Viewing Scheduled Jobs and Logs

View the status of scheduled jobs and view logs for troubleshooting.

Viewing Scheduled Jobs

To view details and status of the scheduled jobs, connect in SQL*Plus as the OAA schema owner, for example `DEV_OAA` and run:

```
SQL> select * from ALL_SCHEDULER_JOBS
```

This will give details such as the job name, last run time, time to execute, and next run time.

Vieweing Scheduler Logs

To view the logs from previous jobs, connect in SQL*Plus as the OAA schema owner, for example `DEV_OAA` and run:

```
SQL> select * USER_SCHEDULER_JOB_LOG
```

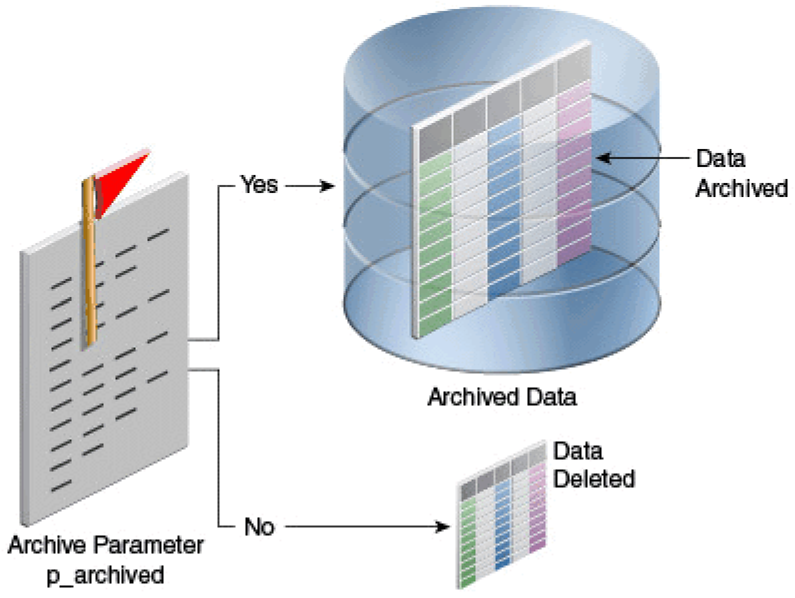
14.3 Archiving and Purging

The archive and purge process allows the releasing of data that is not required anymore for rules evaluation or fraud investigation.

Archiving is the process of moving data from main transactional tables to the archive tables.

Purging is the process of deleting obsolete data that is not required by the system from tables because of data growth. Not all the tables are purged since many of them do not have data growth.

Figure 14-1 Tables Without Data Growth Images Not Purged



"Purging" is different from "backing up data". A data backup is for the recovery of data if loss occurs; purges are for keeping the runtime tables free of old data. Regardless, to protect your data, database backups should be performed on a regular basis with the help of a database administrator.

The following data can be archived or purged using the scripts provided in a zip file inside the management container in the `/u01/oracle/db_purge` directory:

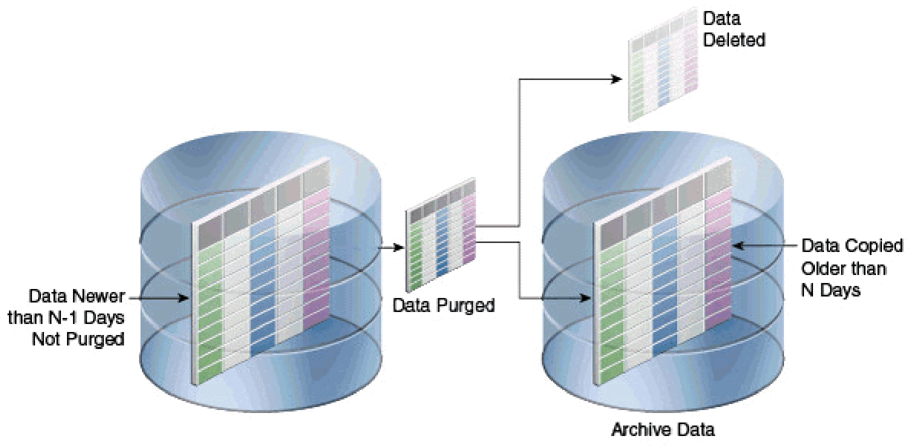
- Login and devices data
- Rule Logs data
- Auto Learning data
- Transactions and Entities data
- Profile data

Archive and purge criteria is based on the create/update timestamp of the records. This is specified using the retention period described using number of days.

The following describes an overview of the archive and purge process:

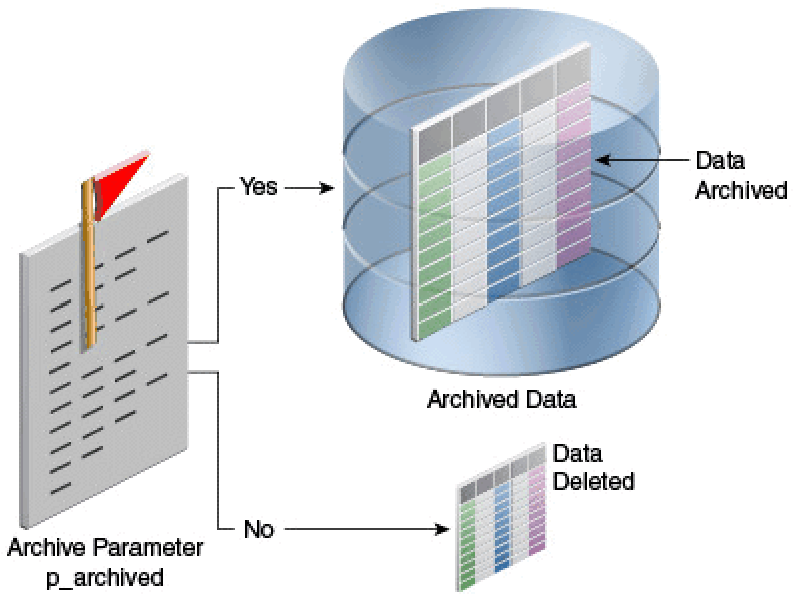
1. Determine the retention period (usually 180 days; that is 6 months).

Figure 14-2 Retention


Data Deleted
Data Newer than N-1 Days Not Purged
Data Purged
Data Copied Older than N Days
Archive Data

2. Determine whether to purge or archive.

Figure 14-3 Determining to Purge or Archive



3. Deploy the purge related stored procedures into the OAA database. This is a one-time job.
4. Determine what types of data must be archived and purged.
5. Schedule the related scripts to run on regular intervals or manually run the scripts when required.
6. Check for entries where the LOG_TYPE is 99 in the database table V_SYS_LOGS.

Note:

Rules may behave differently if the data that they look for is purged. For example, a rule is looking for 6 month data and you are purging data that is 9 days or older.

The following sections describe the archive and purge process in more detail:

- Setting Up the Scripts in the Database
- Running the Archive and Purge Scripts
- Running Partition Maintenance Scripts
- Minimum Data Retention Policy for OLTP (Online Transaction Processing) Tables
- Best Practices/Guidelines for Running Purge Scripts
- Details of Data that is Archived and Purged
- List of Related Stored Procedures

14.3.1 Setting Up the Scripts in the Database

To archive and purge OAA data, you must set up one time scripts as follows:

1. Enter a bash shell for the OAA management container:

   ```
   kubectl exec -n <namespace> -ti <oaamgmt-pod> -- /bin/bash
   ```

   For example:

   ```
   kubectl exec -n oaans -ti oaamgmt-oaa-mgmt-7dfccb7cb7-lj6sv -- /bin/bash
   ```

2. Inside the management container, navigate to the `/u01/oracle/db_purge/archive` directory and login to the OAA database as a `SYSDBA` user. For example:

   ```
   cd /u01/oracle/db_purge/archive
   sqlplus sys/<password>@//db.example.com:1521/orcl.example.com as sysdba
   ```

3. Grant the following privileges to the OAA schema, for example `DEV_OAA`, so that stored procedures can be created and executed:

   ```
   GRANT create any procedure TO <schema_name>;
   GRANT create any table TO <schema_name>;
   GRANT create any index TO <schema_name>;
   GRANT create procedure TO <schema_name>;
   GRANT execute any procedure TO <schema_name>;
   exit;
   ```

4. Login to the database as the OAA schema user, for example `DEV_OAA`:

   ```
   sqlplus <schema_name>/<password>@//db.example.com:1521/orcl.example.com
   ```

5. Run the `create_purge_proc.sql` script to create the purge procedures:

   ```
   SQL> @create_purge_proc.sql
   ```

   When running the `create_purge_proc.sql` script, the script asks for the following inputs:

   ```
   Enter the value for oaam_data_tbs: <schema_name>_TBS_DATA
   Enter the value for oaam_indx_tbs: <schema_name>_TBS_INDX
   ```

6. Validate the stores procedures to make sure they are valid and without errors:

   ```
   SELECT object_name,object_type FROM user_objects WHERE
   status='INVALID' and
         object_type='PROCEDURE';
   ```

Next steps: Running the Archive and Purge Scripts.

14.3.2 Running the Archive and Purge Scripts

To run the archive and purge scripts, proceed as follows:

1. If you are not already inside the mangement container, enter a bash shell for it:

   ```
   kubectl exec -n <namespace> -ti <oaamgmt-pod> -- /bin/bash
   ```

   For example:

   ```
   kubectl exec -n oaans -ti oaamgmt-oaa-mgmt-7dfccb7cb7-lj6sv -- /bin/
   bash
   ```

2. Inside the management container, navigate to the `/u01/oracle/db_purge/archive` directory:

   ```
   cd /u01/oracle/db_purge/archive
   ```

3. Select the scripts to run based on the data that must be archived or purged. The table below lists the types of data and corresponding script name:

   Table 14-1 Archive and Purge Scripts Based on Types of Data

Type of Data	Corresponding Script	Description/Comments
User Authentication related data	exec_sp_purge_tracker_data .sql	All OAA/OARM user authentication activity generates this type of data.
Rules, Policy Log Data	exec_sp_purge_rule_log.sql	All OAA/OARM user authentication activity generates this type of data.
Custom Activities related Data	exec_sp_purge_txn_log.sql	If product is configured to use custom activities then, it generates this type of data.

Table 14-1 (Cont.) Archive and Purge Scripts Based on Types of Data

Type of Data	Corresponding Script	Description/Comments
Behavior pattern related Pattern Data	exec_sp_purge_workflow_data.sql	If product is configured to use profiling / patterns, then it generates this type of data.
User Authentication Profile Data	exec_sp_purge_profile_data.sql	All OAA/OARM user authentication activity generates this type of data.
Data generated from Monitor data points	exec_v_monitor_purge_proc.sql	All OAA/OARM user authentication activity generates this type of data.

4. Edit the sql script you want to run. Set the `p_days1` and `p_archived` parameters accordingly. The table below describes these parameters:

   Table 14-2 Archive and Purge Routine Parameters

Variable Name	Default Value	Description
p_days1	180	Retention period in days. Data older than this many number of days will be archived or purged.
p_archived	Y	Y or N for Yes and No respectively. If "Y" then data will be archived (in archive tables), otherwise data will be purged based on the retention period.

5. Login to the database as the OAA schema user, for example `DEV_OAA` and execute the selected script:

   ```
   sqlplus <schema_name>/<password>@//db.example.com:1521/orcl.example.com
   SQLPLUS> @<script>.sql
   ```

6. Check the corresponding log file and see if there are any errors or warnings.
7. If archiving is selected, then make sure to take a backup of the archive tables so that data can be restored if needed.

Next steps: Running Partition Maintenance Scripts.

14.3.3 Running Partition Maintenance Scripts

As described in Partition Maintenance, when a partitioned OAA database is used, a number of partitions are created by database jobs for some tables.

After completing purging and archiving tasks, administrators can decide which partitions are no longer required. These partitions can then be dropped.

A list of partitions can be found my logging into the database as the OAA schema user and running:

```
SQL> select * from user_tab_partitions
```

Partitions that are no longer required can be dropped for those tables using the `SP_OAA_DROP_PARTITION` procedure. To drop partitions:

1. Login to the database as the OAA schema user.
2. Execute the procedure as follows:

   ```
   SQL> execute SP_OAA_DROP_PARTITION(<table_name>,<p_days>)
   ```

   where:

   - `<table_name>` specifies for which table the partition needs to be dropped
   - `<p_days>` specifies the number of days of data you wish to retain. For example, if you specifiy `p_days` as 180, then all partitions for that table where the data is older than 180 days, will be dropped.

14.3.4 Minimum Data Retention Policy for OLTP (Online Transaction Processing) Tables

Based on the OAA system requirement, the minimum data retention policy for various OLTP (online transaction processing) tables are shown below. Users should determine the data retention period based on their business requirements.

Table 14-3 Minimum Data Retention Policies

Data	Retention Policy
User Authentication related data, User Authentication Profile Data	Minimum of 6 months or 180 days
Data generated from Monitor data points	Minimum of 6 months or 180 days
Custom Activities related Data	Data that has not been updated in the last 180 days is purged by default.
Behavior pattern related Pattern Data	Retention for hours, days, months, and years is listed below: • HOURS based pattern-workflow tables will retain 3 days worth of data. • DAYS based pattern-workflow tables will retain 32 days worth of data. • MONTHS based pattern-workflow tables will retain 1 years worth of data. • YEARS based pattern-workflow tables will retain 5 years worth of data.
Rule Log Data	The archive and purge script will archive and purge all rule log data that is 30 days older (This value should be set based on the customer care requirement. If the reporting database is used, then, rule logging data retention should be less than 30 days.

14.3.5 Best Practices/Guidelines for Running Purge Scripts

The following is a list of best practices and guidelines for running purge scripts:

- Determine the retention period based on the business requirements and rules and policies used.
- Perform regular purge/archive.
- Make sure replication is not enabled during the window when these scripts are run.
- Run the scripts during off peak load hours as archive and purge can be resource (CPU) intensive.
- If archiving is required, make sure there is enough disk space available on the database server since the data would be moved to archive tables instead of simply purging. Archival space should be equal to or greater than the current table's storage.
- Plan your purging strategy as purging requires a significant amount of time if there are millions of rows that need to be deleted or copied from the database.
- Oracle recommends that custom purging scripts only include the tables used by the standard purging scripts provided. The alterations to the provided purge scripts can include parameterization for user ID. Such alterations should be thoroughly tested before being used in production to ensure they function as expected.

14.3.6 Details of Data that is Archived and Purged

Details of data that is purged and the corresponding archived tables, are presented below:

Table 14-4 User Authentication related Data

User Authentication related Tables	Corresponding Archived Tables
VCRYPT_TRACKER_NODE	VCRYPT_TRACKER_NODE_PURGE
VCRYPT_TRACKER_NODE_HISTORY	VCRYPT_TRACKER_NODE_HISTORY_PURGE
VCRYPT_TRACKER_USERNODE_LOGS	VCRYPT_TRACKER_USERNODE_LOGS_PURGE
VT_DYN_ACT_EXEC_LOG	VT_DYN_ACT_EXEC_LOG_PURGE
VT_SESSION_ACTION_MAP	VT_SESSION_ACTION_MAP_PURGE
VT_USER_DEVICE_MAP	VT_USER_DEVICE_MAP_PURGE
VCRYPT_ALERT	VCRYPT_ALERT_PURGE
VCRYPT_USERS_HIST	VCRYPT_USERS_HIST_PURGE
V_USER_QA_HIST	V_USER_QA_HIST_PURGE

Table 14-5 Rules and Policy Log Data

Rules, Policy Log Tables	Corresponding Archived Tables
VR_POLICYSET_LOGS	VR_POLICYSET_LOGS_PURGE
VR_RULE_LOGS	VR_RULE_LOGS_PURGE
VR_MODEL_LOGS	VR_MODEL_LOGS_PURGE
VR_POLICY_LOGS	VR_POLICY_LOGS_PURGE

Table 14-6 Custom Activities related Data

Transaction Tables	Corresponding Archived Tables
VT_ENTITY_ONE	VT_ENTITY_ONE_PURGE

Table 14-6 (Cont.) Custom Activities related Data

Transaction Tables	Corresponding Archived Tables
VT_ENTITY_ONE_PROFILE	VT_ENTITY_ONE_PROFILE_PURGE
VT_USER_ENTITY1_MAP	VT_USER_ENTITY1_MAP_PURGE
VT_ENT_TRX_MAP	VT_ENT_TRX_MAP_PURGE
VT_TRX_DATA	VT_TRX_DATA_PURGE
VT_TRX_LOGS	VT_TRX_LOGS_PURGE

Table 14-7 Behavior Pattern Related Data

Autolearning Transactional Tables	Corresponding Archived Tables
VT_WF_DAYS	VT_WF_DAYS_PURGE
VT_WF_HOURS	VT_WF_HOURS_PURGE
VT_WF_MONTHS	VT_WF_MONTHS_PURGE
VT_WF_YEARS	VT_WF_YEARS_PURGE
V_FPRINTS	V_FPRINTS_PURGE
V_FP_MAP	V_FP_MAP_PURGE

Table 14-8 User Authentication Profile Data

Tranasactional Tables	Corresponding Archived Tables
VT_USER_PROFILE	VT_USER_PROFILE_PURGE
VT_DEVICE_PROFILE	VT_DEVICE_PROFILE_PURGE
VT_BASE_IP_PROFILE	VT_BASE_IP_PROFILE_PURGE
VT_IP_PROFILE	VT_IP_PROFILE_PURGE
VT_STATE_PROFILE	VT_STATE_PROFILE_PURGE
VT_CITY_PROFILE	VT_CITY_PROFILE_PURGE
VT_COUNTRY_PROFILE	VT_COUNTRY_PROFILE_PURGE

Table 14-9 Monitor Data

Transaction Table	Corresponding Archived Tables
V_MONITOR_DATA	V_MONITOR_DATA_PURGE

14.3.7 List of Related Stored Procedures

The `create_purge_proc.sql` script creates the tables and the following stored procedures to archive and purge data from the transaction tables:

- SP_RULE_PROC
- SP_MODEL_PROC
- SP_POLICYSET_PROC
- SP_POLICY_PROC
- SP_NODE_HISTORY_PROC

- SP_NODE_PROC
- SP_USER_NODE_PROC
- SP_USER_DVC_PROC
- SP_SESS_ACT_MAP_PROC
- SP_WF_YEARS_PROC
- SP_WF_MONTHS_PROC
- SP_WF_DAYS_PROC
- SP_WF_HOURS_PROC
- SP_V_FPRINTS_PROC
- SP_V_FP_MAP_PROC
- SP_VT_DY_ACT_EX_LOG_PRO
- SP_VT_TRX_LOGS_PROC
- SP_VT_TRX_DATA_PROC
- SP_VT_ENT_TRX_MAP_PROC
- SP_VT_ENT_ONE_PRF_PROC
- SP_VT_ENT_ONE_PROC
- SP_VT_ENT_ONE_MAP_PROC
- SP_VT_USER_PRF_PROC
- SP_VT_DEVICE_PRF_PROC
- SP_VT_IP_PRF_PROC
- SP_VT_BASE_IP_PRF_PROC
- SP_VT_CITY_PRF_PROC
- SP_VT_COUNTRY_PRF_PROC
- SP_VT_STATE_PRF_PROC
- SP_ARCHIVE_PURGE_VCRYPT_ALERT
- SP_ARCHPURGE_VCRYPTUSERSHIST
- SP_ARCH_PURGE_V_USER_QA_HIST

15 Accessibility Features and Tip

Currently, there are no accessibility features in Oracle Advanced Authentication (OAA). However, you can use the following accessibility tip in the OAA user interface:

Navigating Through the UI Using Keyboard

You can navigate through the elements of the OAA Admin Console using keyboard. For example, in the **Assurance Levels** page under **Uses**, you can navigate in the following way:

1. Navigate through each of the options on the page using the Tab key.
2. Under **Use the Factor(s)**, select the factors check-boxes, for example, Oracle Mobile Authenticator, using the Space key.
3. Navigate to the Validate button using Tab and tap the Enter key.

Part VI
Managing Oracle Adaptive Risk Management

OARM provides a streamlined and a robust interface for administrators and analysts. Administrators can easily identify access requests and monitor alerts to uncover fraud and misuse. This information is easily captured for use and to influence future real-time risk analysis.

This chapter includes the following section:

- Typical OARM Use Cases
- Device Fingerprinting and Identification

16
Typical OARM Use Cases

OARM offers a streamlined and a robust interface for Administrators to proactively determine the risk of an access request and to configure the appropriate outcomes to prevent any fraud or misuse.

Topics

You can use the OARM out-of-the-box offerings to perform a wide range of tasks. This section describes the following example use cases that you can configure in OARM:

- Configuring a Risky IP Use Case
- Configuring a Geo-Velocity Based Use Case
- Loading Geo-Location Data

See the Configuring a Custom Activity Use Case in Oracle Adaptive Risk Management tutorial for details on how to use a custom activity in OARM.

16.1 Configuring a Risky IP Use Case

IP address is one of the most significant data point that Administrators analyze to take prompt action to prevent any fraudulent user activity.

This use case considers a scenario where the Administrator wants to configure IP addresses that are considered as risky for the organization. This use case is achieved by using the **Challenge based on Risky IP** out-of-the-box rule. The outcome of configuring this rule is to raise a risk-based challenge for the user and to generate an alert for the user activity for logins from the IP address that is considered as risky. The Administrator can monitor alerts, actions, rules, and other user-related information through the User Session dashboard. The steps in this use case are also shown in the tutorial Configuring a Risky IP Use Case in Oracle Adaptive Risk Management.
To configure this use case, perform the following steps:

1. Log in to the OARM Administration console.
2. Click the Application Navigation icon to display the left pane, and then click **Adaptive Risk Management**.

   The User Activity dashboard appears.
3. From the User Authentication tile, click the **Rules** link.

   The User Activity rules display page appears.
4. In the search pane, enter the relevant text to filter all the rules available out-of-the-box to configure risky IP, for instance, `risky ip`.

   Challenge based on Risky IP rule appears that you need to configure for this use case.
5. Click the **Edit** icon against the Challenge based on Risky IP rule.

> **Note:**
>
> The Challenge based on Risky IP out-of-the-box rule has a condition associated that evaluates the risky IP address.

6. Verify that the **Select Action** and the **Select Alert** lists are pre-populated with **Challenge** and **Risky IP** options respectively.

> **Note:**
>
> You can configure action and alert as per your requirement. For instance, if the access request is from an IP address that is considered risky and you want to block the user, then you can configure the action as **Block**.

7. Add the risky IP addresses in a group. For the convenience of the Administrator, **Risky IPs** group is provided out-of-the-box.
8. Under IP Group, with **Risky IPs** option selected in the list, click the **Edit Risky IPs** link to add the IP addresses considered as risky.
9. Click **Save and Proceed**.

   The Edit Group page appears.
10. Perform the following steps to configure the Risky IPs group:
    a. Click **Add IPs**.
    b. In the **Value** field, enter the IP address. For instance, 192.0.2.1.
    c. Click **Add**.
    d. Repeat steps 10a to 10c to add the list of risky IP addresses in the group.
11. Click **Save** to save the group.

    You are redirected to the Edit rule page.
12. Click **Save** to save the rule.

    You are redirected to the User Activity rules page.

Now, during the authentication flow when this rule is executed the condition associated with the Risky IP out-of-the-box rule is evaluated. If this condition is evaluated to **True** , then the rule is triggered. In turn, the user is presented the challenge based on the factors configured.

> **Note:**
>
> To learn how to configure factors, see Managing Factors in the User Preferences UI.

16.2 Configuring a Geo-Velocity Based Use Case

OARM allows you to configure geo-velocity as a rule that grants an added layer of security and consequently a higher level of protection to an organization.

Geo-velocity rule allows you to authenticate a user based on the distance and the time gap between your current location and where you last logged in from. You can leverage this information as a criteria for granting access to the protected resource.
Geo-velocity is usually calculated as maximum miles-per-hour. This allows you to determine how fast a user can travel from one place to another to successfully sign in within a specific time duration.

A pre-requisite to implement the geo-velocity use case is it to have the geo-location data. The geo-location feature allows you to identify the physical location of the user. This is usually determined by obtaining the IP address of the device being used by a user to attempt a login. This data is then used to calculate the distance between two consecutive login attempts.

It is possible for a user to log in to an application from a device, then take a flight to another country, and once again log in to the same application using the same device. However, if the calculated velocity is greater than the configured velocity, then an appropriate action and an alert is triggered. Consider a scenario, where a user logs in from India at 9 am (IST), and then two hours later again tries to login from Australia at 11 am (IST). Even with the fastest mode of transportation, the user cannot travel this distance in two hours. It is a clear indication that two different people are trying to log in. This indicates a fraudulent user activity and requires an appropriate action.

The Administrator can use the **Challenge based on Device Maximum Velocity** out-of-the-box rule to detect such type of fraudulent user activity, trigger an alert, and challenge the user from successfully signing in. This is accomplished in conjunction with the geo-location data. The Administrator can monitor and view these alerts, actions, rules, and other user-related information through the **Monitor User Sessions** dashboard.

How the Rule Works

The Device Maximum Velocity rule has two values that the Administrator can configure to calculate the geo-velocity before the rule is triggered. Those value fields are called **Last login within (Seconds)** and **Miles Per Hour is more than**. Using these two field values you can customize the geo-velocity that a physical device can travel before an alert is triggered.

You must bear in mind while setting the Device Maximum Velocity that you cannot change one of the preceding values without considering that the other needs to be updated as well. In other words, you cannot only set the **Last login within (Seconds)** value and not properly adjust the **Miles Per Hour is more than** value. These two values work in conjunction to calculate the device velocity. The relationship between these two settings is an AND.

Let us see how the rule works.

1. The rule first obtains the last successful login within (Seconds).
2. The rule then obtains the last login city and the current login city to calculate the distance between them.
3. The calculated distance between the two cities divided by the time difference in the login times is used to calculate the velocity.
4. If the calculated velocity is greater than the configured velocity, the rule triggers.

> **Note:**
>
> Assumptions to implement this rule are as follows:
>
> - The geo-location data must have been loaded in the OARM server. See Loading Geo-Location Data.
> - The user must login from the same device.
> - The authentication status of the user is successful in the previous login (N seconds ago).

To configure this use case, perform the following steps:

> **Note:**
>
> The steps in this use case are also shown in the tutorial Configuring a Geo-Velocity Based Use Case in Oracle Adaptive Risk Management.

1. Log in to the OARM Administration console.
2. Click the Application Navigation icon to display the left pane, and then click **Adaptive Risk Management**.

   The User Activity dashboard appears.
3. From the **User Authentication** tile, click the **Rules** link.

   The User Activity rules display page appears.
4. In the search pane, enter the relevant text to filter all the rules available out-of-the-box to configure geo-velocity.

   Challenge based on Device Maximum Velocity rule appears that you need to configure for this use case.
5. Click the **Edit** icon against the Challenge based on Device Maximum Velocity rule.

   > **Note:**
   >
   > The Challenge based on Device Maximum Velocity out-of-the-box rule has an associated condition that evaluates the maximum velocity of the device in the specified time.
6. Verify that the **Select Action** and the **Select Alert** lists are pre-populated with **Challenge** and **Device Maximum Velocity** options respectively.

   > **Note:**
   >
   > You can configure action and alert as per your requirement.
7. Verify that the **Last login within (Seconds)** and **Miles per Hour is more than** fields are pre-populated with **72000** and **600** respectively.

> **Note:**
>
> You can configure the preceding fields as per your requirement.

8. Add the IP addresses that you want to ignore for the Device Maximum Velocity rule. For the convenience of the Administrator, **Ignore IP Group** group is provided out-of-the-box.

   > **Note:**
   >
   > This parameter allows you to specify a list of IPs to ignore. If the IP of the user is from that list, then this condition always evaluates to false. If the IP of the user is not in that list or if the list is null or empty, then the condition evaluates the velocity of the user or the device from the last login and evaluates to true if the velocity exceeds the configured value.

9. Under Ignore IP Group, with **Ignore IP Group** option selected in the list, click the **Edit Ignore IP Group** link to add the IP addresses to ignore for this rule.
10. Click **Save and Proceed** .

    Edit Ignore IP Group page appears.
11. Perform the following steps to configure the group:
    a. Click **Add IPs**.
    b. In the **Value** field, enter the IP address. For instance, 192.0.2.1.
    c. Click **Add**.
    d. Repeat Steps 11 a to 11 c to add the list of IP addresses to ignore in the group.
12. Click **Save** to save the group.

    You are redirected to the Edit rule page.
13. Click **Save** to save the rule.

    You are redirected to the User Activity rules page.

Now, during the authentication flow when this rule is executed the condition associated with the Device Maximum Velocity out-of-the-box rule is evaluated. If this condition is evaluated to **True** , then the rule is triggered. In turn, the user is presented the challenge based on the factors configured.

> **Note:**
>
> To learn how to configure factors, see Managing Factors in the User Preferences UI.

16.3 Loading Geo-Location Data

OARM leverages geo-location data for detecting fraudulent user activity and reporting.

Geo-location data helps you identify the physical location of the user. Geo-location data denotes the location information by obtaining the IP addresses of the user. Consequently, you can detect where the fraudulent user activity has occurred to take immediate action.

OARM supports IP geo-location data from the following providers:

- Neustar Version 7
- Neustar (formerly Quova)
- Maxmind

The OAA management container is typically used to load IP geo-location data into the OARM database schema. You can, however, load geo-location data from outside the OAA management container also.

Prerequisite: You must ensure that OARM is installed and running, before you perform the steps to load geo-location data.

> **Note:**
>
> - Loading geo-location data can be time consuming, but it happens in the background while the service continues to function.
> - If you have enabled archival logs, make sure you back them up periodically (at least every half an hour) and that the backed up logs are purged.

Loading Geo-location Data from Within the OAA Management Container

Perform the following steps:

1. Download the geo-location data to a working directory of your choice, for instance, `$WORKDIR/geoData`.
2. Copy the geo-location data to the NFS volume `<NFS_VAULT_PATH>`, so that it can be accessed by the OAA management container.

```
$ cd <NFS_VAULT_PATH>
$ mkdir -p geoData
$ sudo cp $WORKDIR/geoData/*.* <NFS_VAULT_PATH>/geoData
```

> **Note:**
>
> You can copy the data files in any location inside the <NFS_VAULT_PATH>. It is not mandatory to place it under the geoData folder.

3. Set the files permissions as follows:

   ```
   $ cd <NFS_VAULT_PATH>/geoData
   $ chmod 444 *.*
   ```

4. Enter a bash shell for the OAA management pod if not already inside one:

   ```
   kubectl exec -n <namespace> -ti <oaamgmt-pod> -- /bin/bash
   ```

5. Ensure that the geo-location files are visible inside the management container:

   ```
   ls -l /u01/oracle/service/store/oaa/geoData
   ```

   For example, for Neustar version 7 the files will look similar to the following:

   ```
   --r--r--r-- 1 oracle staff 3673477337 Jan 26 15:22 
   oracletest_cgp_v1133.csv.gz
   ```

6. Navigate to the `/u01/oracle/oaa_cli/bharosa_properties` directory inside the container.

   ```
   cd /u01/oracle/oaa_cli/bharosa_properties
   ```

7. Edit the `bharosa_location.properties` file to reflect the location data provider and the location of data files.
   For Neustar, IP location loader related properties are defined here:

   ```
   ### IP location loader specific properties go here

   ### Specify the data provider:  neustarV7 or maxmind or quova(for quova 
   legacy format)
   location.data.provider=neustarV7

   ### Specify the data file, for neustarV7 or maxmind or quova(for quova 
   legacy format)
   location.data.file=/u01/oracle/service/store/oaa/geoData/
   test_cgp_v1114.csv.gz

   ### Specify the reference file for quova (for data provided by quova/
   neustar in legacy format).For NeustarV7, this property can be commented 
   (optional).
   location.data.ref.file=/u01/oracle/service/store/oaa/geoData/
   test_08132006.ref.gz

   ### Specify the anonymizer data file for quova (for data provided by 
   quova/neustar in legacy format).For NeustarV7, this property can be 
   commented (optional).
   location.data.anonymizer.file=/u01/oracle/service/store/oaa/geoData/
   test_anonymizer.dat.gz
   ```

For Maxmind, IP location loader related properties are defined here:

```
### Specify the data provider: maxmind or quova (for data provided
by neustar)
location.data.provider=maxmind

### Specify the location data file, for maxmind
location.data.location.file=/u01/oracle/service/store/oaa/geoData/
GeoIP2-Enterprise-Locations-en.CSV

### Specify the blocks data file, for maxmind
location.data.blocks.file=/u01/oracle/service/store/oaa/geoData/
GeoIP2-Enterprise-Blocks-IPv4.CSV

### Specify the country code data file, for maxmind
location.data.country.code.file=/u01/oracle/service/store/oaa/
geoData/ISO_3166_CountryCode.csv

### Specify the sub country code data file, for maxmind
location.data.sub.country.code.file=/u01/oracle/service/store/oaa/
geoData/FIPS_10_4_SubCountryCode.csv
```

Note:

Regardless of whether you are using Neustar or Maxmind, leave all the values uncommented. For example, if using Neustar and you set the Neustar properties accordingly, you must leave the Maxmind properties uncommented even though they are not being used.

Oracle recommends using the default values for the remaining parameters:

```
### Specify the number of database threads
location.loader.database.pool.size=16

### Specify the maximum number of location records to batch before
issuing a database commit
location.loader.database.commit.batch.size=100

### Specify the maximum time to hold an uncommitted batch
location.loader.database.commit.batch.seconds=30

### Specify the maximum number of location records to be kept in
queue for database threads
location.loader.dbqueue.maxsize=5000

### Specify the maximum number of location records to be kept in
cache
location.loader.cache.location.maxcount=5000

### Specify the maximum number of location split records to be kept
in cache
location.loader.cache.split.maxcount=5000
```

```
### Specify the maximum number of anonymizer records to be kept in cache
location.loader.cache.anonymizer.maxcount=5000

### Specify the maximum number of ISP records to be kept in cache
location.loader.cache.isp.maxcount=5000
```

8. Load the data by running the `loadIPLocationData.sh` script.

   ```
   cd /u01/oracle/oaa_cli
   ./loadIPLocationData.sh
   ```

   Note:

   When running this script, the properties file can be passed as an optional parameter as follows:

   ```
   cd /u01/oracle/oaa_cli
   ./loadIPLocationData.sh -f ./../scripts/settings/
   installOAA.properties
   ```

   The preceding file parameter (-f) is optional. If this value is not provided, then the property file's default value is fetched from `/u01/oracle/scripts/settings/installOAA.properties`.

   You can override this file value by setting the environmental variable `INSTALL_PROP_FILE` in `setCliEnv.sh`. The file must contain the following information:

   - `database.host`=<Database Host Name>
   - `database.port`=<Database Port Number>
   - `database.schema`=<Database User Name>
   - `database.schemapassword`=<Database Schema Password>: This property is optional.
   - `database.svc`=<Database Service Name>
   - `database.name`=<Database Name>

   You must keep in mind that one of the two options, `database.svc` or `database.name`, must be present. If both are present in the file, the `database.svc` value takes precedence.

   Note:

   Enter the password to the schema if prompted.

   The data can take several hours to load.

Loading Geo-location Data from Outside the OAA Management Container

Perform the following steps:

1. Copy recursively all files and folders under OAA management container's `<BHCLI_HOME>` folder to the new location on the target compute node from where you intend to run the geo-location loading script.

   > **Note:**
   >
   > The environmental variable `<BHCLI_HOME>` is set to indicate the loader home folder. The value is `/u01/oracle/oaa_cli` by default.

2. Check that the data file to be loaded is located at the path specified in `<BHCLI_HOME>/bharosa_properties/bharosa_location.properties`. Update the properties file as necessary.
3. Set the following environment variable to the desired location to create a log file.

   ```
   LOGS_DIR=/public/geoDatLogs/logs
   ```

4. Install Java on the external compute and change the `JAVA_HOME` environmental variable to the management container's Java version.

   > **Note:**
   >
   > You must ensure that the `installOAA.properties` file from the management container is either visible or copied over to the external compute. Perform the load, providing parameters as needed.

5. Load the data by running the `loadIPLocationData.sh` script.

17 Device Fingerprinting and Identification

Device fingerprinting/identification is one of the many attributes OARM uses to assess the risk of an access request or transaction.

Whether it is a desktop computer, laptop computer, mobile device, or other web-enabled device, OARM can use any combination of standard attributes, such as browser user agent string data, proprietary secure cookies, and advanced Autolearning device identification logic, to identify a device. This chapter covers the important fingerprinting and identification, concepts, and technology customers need to understand when deploying OARM.

> **Note:**
>
> Positive device identification is not and should not be considered an authentication method, nor the sole determining factor of an allow or block decision. OAA and OARM provides a full, layered security solution. Device fingerprinting and identification represents only one of the layers.

Topics

- Overview of Device Fingerprinting

17.1 Overview of Device Fingerprinting

OARM device fingerprinting is a capability used to recognize the devices a user uses to login and conduct transactions. It collects information about the device like browser type, browser headers, operating system type, locale, and so on. Fingerprint data represents the data collected for a device during the login process, which is required to identify the device whenever it logs in the next time. The fingerprint details help in identifying whether a device is secure and determine the risk level of the authentication or transaction.

A device is identified using proprietary logic and a set of specialized policies to process available data and arrive at identification. The intelligent identification does not rely on any single attribute type so it can function on user devices not following strict specifications and in both web and non-web channels. The device identification is not merely a static list of attributes but is instead a dynamic capture, evaluation and profiling of the specific combinations of attributes available in each access request or transaction. This is especially important in large consumer facing deployments.

This section includes the following topics:

- Fingerprinting Types
- **What Makes Up a Device Fingerprint?**

17.1.1 Fingerprinting Types

As standard, OARM supports browser and JavaScript fingerprints. The fingerprinting functions the same for desktop/laptop PCs and mobile devices and smart phones that run full-function browsers.

Web Browser-Based Fingerprinting

By design OARM provides web browser based fingerprinting in a pure web environment. In other words, no client software is required, which makes deployment of the solution to large and diverse user populations manageable. Also, OARM does not place any logic on the client side where it may be vulnerable to exploit.

When an end user is accessing a protected application via a web browser, OARM performs browser based fingerprinting. Browser based fingerprinting and identification uses browser user-agent string data and secure cookie data if available.

JavaScript Fingerprinting

OARM provides fingerprinting with JavaScript.

17.1.2 What Makes Up a Device Fingerprint?

The overall fingerprinting of a user device is based on multiple factors which is explained in this section.

OARM's fingerprinting technology does not solely rely on one element. OARM uses dozens of attributes to recognize and fingerprint the device typically used to login, providing greater coverage. For example, where certain elements are unavailable, the system can still provide robust security utilizing other objects, such as secure cookie or HTTP headers.

Secure Cookie and Browser Characteristics

Secure cookies are one of the attributes used to identify the device. OARM generates a unique Secure Cookie for each identification and looks for the same cookie the next time any user logs in from the device. The cookie is only valid for that session on that particular device. If the end user logs out and logs back in, that cookie is used to identify the device at that point.

Note:

If there is a policy that does not allow cookies, the secure cookie will not persist.

The Secure Cookie is extracted from the HTTP request. Along with the secure cookie, OARM also extracts browser characteristics.

For additional characteristics that are used to create a unique fingerprint for the device, refer to the browser fingerprint enum and table below:

OS/Browser	Characteristics
Operating System	• Operating System • Version • Patch level
Browser	• Browser • Version • Patch level
Locale	• Country • Language • Variant

The browser fingerprint type enum is shown below to illustrate the information to be collected for a browser fingerprint:

```
#Enum for fingerprint type
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,local
variant,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,Englis
h,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,Hungar
ian,no,Norwegian,pl,Polish,pt,Portuguese,ro,Romanian,ru,Russian,sk,Slovak,sv,
Swedish,th,Thai,tr,Turkish,BR,Brazil
```

JavaScript and Device Characteristics

OARM also provides fingerprinting with JavaScript.

The JavaScript fingerprint type enum is shown below to illustrate the information to be collected for a JavaScript fingerprint:

```
vcrypt.fingerprint.type.enum.javascript.header_list=acn,gl,amv,l,ce,an,av,p,u
a,o,je,te,w,h,cd,aw,ah,tzo,mt,pl,osc,prod,prods,bid,pd,cc,dnt
vcrypt.fingerprint.type.enum.javascript.cc=CPU class
vcrypt.fingerprint.type.enum.javascript.cd=Color depth
vcrypt.fingerprint.type.enum.javascript.dnt=Do not track
vcrypt.fingerprint.type.enum.javascript.ce=Cookies enabled
vcrypt.fingerprint.type.enum.javascript.tzo=Timezone offset
vcrypt.fingerprint.type.enum.javascript.result_list=acn,l,ua
vcrypt.fingerprint.type.enum.javascript.is_device_fingerprint=true
vcrypt.fingerprint.type.enum.javascript.gl=Location
vcrypt.fingerprint.type.enum.javascript.mt=Mime types
vcrypt.fingerprint.type.enum.javascript.ah=Available height
vcrypt.fingerprint.type.enum.javascript.prods=Sub Product
vcrypt.fingerprint.type.enum.javascript.header_name_nv=acn,App code
name,gl,Location,amv,App minor version,l,Language,ce,Cookies
enabled,an,App name,av,App version,p,Platform,ua,User
agent,o,Online,je,Java enabled,te,Taint
enabled,w,Width,h,Height,cd,Color depth,aw,Available
width,ah,Available height,tzo,Timezone offset,mt,Mime
types,pl,Plugins,osc,OS CPU,prod,Product,prods,Sub product,bid,Build
ID,pd,Pixel depth,cc,CPU class,dnt,Do not track
vcrypt.fingerprint.type.enum.javascript.an=App name
vcrypt.fingerprint.type.enum.javascript.name=Javascript
vcrypt.fingerprint.type.enum.javascript.prod=Product
vcrypt.fingerprint.type.enum.javascript.te=Taint enabled
vcrypt.fingerprint.type.enum.javascript.description=Javascript
vcrypt.fingerprint.type.enum.javascript.pd=Pixel depth
vcrypt.fingerprint.type.enum.javascript.osc=OS CPU
vcrypt.fingerprint.type.enum.javascript.search_list=acn,l,ua
vcrypt.fingerprint.type.enum.javascript.av=App version
vcrypt.fingerprint.type.enum.javascript.header_value_nv=t,true,f,false,
en,English,es,Spanish,de,German,it,Italian,ja,Japanese,fr,French,ko,Kor
ean,zh,Chinese,ar,Arabic,cs,Czech,da,Danish,nl,Dutch,fi,Finnish,el,Gree
k,iw,Hebrew,hu,Hungarian,no,Norwegian,pl,Polish,pt,Portuguese,ro,Romani
an,ru,Russian,sk,Slovak,sv,Swedish,th,Thai,tr,Turkish,BR,Brazil,CA,Cana
da
vcrypt.fingerprint.type.enum.javascript.aw=Available width
vcrypt.fingerprint.type.enum.javascript.bid=Build ID
vcrypt.fingerprint.type.enum.javascript.je=Java enabled
vcrypt.fingerprint.type.enum.javascript.pl=Plugins
vcrypt.fingerprint.type.enum.javascript=4
vcrypt.fingerprint.type.enum.javascript.processor=oracle.security.uas.c
ore.uio.processor.device.JSDeviceIdentificationProcessor
vcrypt.fingerprint.type.enum.javascript.amv=App minor version
vcrypt.fingerprint.type.enum.javascript.acn=App code name
vcrypt.fingerprint.type.enum.javascript.p=Platform
vcrypt.fingerprint.type.enum.javascript.ua=User agent
vcrypt.fingerprint.type.enum.javascript.w=Width
vcrypt.fingerprint.type.enum.javascript.h=Height
vcrypt.fingerprint.type.enum.javascript.l=Language
vcrypt.fingerprint.type.enum.javascript.o=Online`
```

Part VII
Appendices

Topics

A
Understanding OAA/OARM Schema Reference

OAA/OARM provides you access to a rich set of forensic data to generate custom reports for investigation and analysis.

To query and generate reports on information in the OAA/OARM database schema, you can use any reporting solution, such as Oracle Business Intelligence (BI) Publisher.

This chapter contains in-depth information on database tables. It contains the following sections:

- Viewing the Details of Database Tables
- Using Geo-Location Data
- Building OAA/OARM Custom User Activity Reports
- Creating Custom Report Example

A.1 Viewing the Details of Database Tables

Learn about the specifics of each database table.

Topics

- VCRYPT_USER_GROUPS
- VCRYPT_TRACKER_USERNODE_LOGS
- VCRYPT_TRACKER_NODE
- VT_USER_DEVICE_MAP
- VT_SESSION_ACTION_MAP
- VT_USER_GROUPS
- V_FPRINTS
- V_FP_NV
- V_FP_MAP
- VCRYPT_COUNTRY
- VCRYPT_STATE
- VCRYPT_CITY
- VCRYPT_ISP
- VCRYPT_IP_LOCATION_MAP
- VT_TRX_DEF
- VT_TRX_INPUT_DEF
- VT_ENTITY_DEF

- VT_TRX_ENT_DEFS_MAP
- VT_ENT_DEFS_MAP
- VT_DATA_DEF
- VT_DATA_DEF_ELEM
- VT_DATA_DEF_MAP
- VT_DATA_DEF_TRANS
- VT_ELEM_DEF_TRANS
- VT_TRANS_SRC_ELEM
- VT_TRX_LOGS
- VT_TRX_DATA

A.1.1 VCRYPT_USER_GROUPS

Discover the specifics of the `VCRYPT_USER_GROUPS` database table.

Description: This table contains the user group details.

Database table name: `VCRYPT_USER_GROUPS`

Primary Key: *`GROUP_ID`*

Database Column Name	Database Column Type	Description	Length	Enum Values
GROUP_ID (PK)	BIGINT	Id for the User group	16	-
GROUP_NAME	TEXT	Name of the group	4000	-
DESCRIPTION	TEXT	Description for this group	4000	-
CREATE_TIME	DATETIME	Date/time creation of this user	6	-
UPDATE_TIME	TIMESTAMP	Last update time	6	-
USERGROUP_TYPE_CODE	INT	Type of the User group	-	-
USERGROUP_STATUS_CODE	INT	Status of the User group	2	• STATUS_ACTIVE • STATUS_DISABLED • STATUS_DELETED
NOTES	TEXT	Note	4000	-

A.1.2 VCRYPT_TRACKER_USERNODE_LOGS

Discover the specifics of the `VCRYPT_TRACKER_USERNODE_LOGS` database table.

Description: This table logs all the activities for the nodes.

Database table name: `VCRYPT_TRACKER_USERNODE_LOGS`

Primary Key: *`USER_NODE_LOG_ID`*

Database Column Name	Database Column Type	Description	Length
USER_NODE_LOG_ID (PK)	BIGINT	Log Id of the user for a given node	16
REQUEST_ID	TEXT	Id of the request. This is used to co-relate the post authentication request.	256
EXT_SESSION_ID	TEXT	External session Id	512
CLIENT_DEVICE_ID	TEXT	Id of the device which is generated by the application.	256
REMOTE_IP_ADDR	BIGINT	The IP address from where the client connected	15
BASE_IP_ADDR	BIGINT	This is the base IP address for quick search	15
NODE_ID	BIGINT	Id of the nodeId.	16
TRACKER_NODE_HISTORY_ID	BIGINT	Id of the Tracker Node History (if available).	16
USER_ID	TEXT	User Id of the user if available.	256
USER_LOGIN_ID	TEXT	Login Id of the user if available.	256
USER_GROUP_ID	TEXT	GroupId of the user if available.	256
USER_SUB_GROUP_ID	TEXT	Sub GroupId of the user if available.	256
AUTH_STATUS	INT	Status of the authentication.	3
CREATE_TIME	DATETIME	Date/time for this log.	6
UPDATE_TIME	TIMESTAMP	Last update time for this object.	6
EXEC_TIME	TIMESTAMP	The time when this request was processed.	6
IS_REGISTERED	CHAR	Whether this node is registered.	-
SENT_DIG_SIG_COOKIE	VARCHAR	Digital signature cookie that was sent by the UI	128
EXPECTED_DIG_SIG_COOKIE	VARCHAR	Digital signature cookie that was expected by the server from the UI for this node	128
SENT_SECURE_COOKIE	VARCHAR	Secure cookie that was sent by the UI	128
EXPECTED_SECURE_COOKIE	VARCHAR	The secure cookie that was expected by the server from the UI for this node	128

Database Column Name	Database Column Type	Description	Length
AUTH_CLIENT_TYPE_CODE	INT	Type of the client used by the user for authentication	2
CLIENT_VERSION	VARCHAR	Version of the client used for authentication	24
DIGITAL_CLIENT_TYPE_CODE	INT	Type of the client used by the digital cookie client	2
DIGITAL_CLIENT_VERSION	VARCHAR	Version of the client used by the digital cookie client	24
SECURE_CLIENT_TYPE_CODE	INT	Type of the client used by the secure cookie client	2
SECURE_CLIENT_VERSION	VARCHAR	Version of the client used by the secure cookie client	24
DIGITAL_FP_ID	BIGINT	Fingerprint Id of the digital cookie request	16
FPRINT_ID	BIGINT	Log Id for the fingerprint	16
LOAD_DURATION	INT	Time taken to load the page	8
DEVICE_SCORE	INT	Score for the device for this login	8
PREAUTH_SCORE	INT	Pre Authentication score	8
POST_SCORE	INT	Post Authentication score	8
PREAUTH_ACTION	TEXT	Pre Authentication action	256
POST_ACTION	TEXT	Post Authentication action	256
CITY_SCORE	INT	Score for the city for this login	8
STATE_SCORE	INT	Score for the state for this login	8
COUNTRY_SCORE	INT	Score for the country for this login	8
POST_PROCESS_STATUS	INT	Status of the post processing	5
POST_PROCESS_RESULT	INT	Result of the post processing	5
LOGIN_FLAG	INT	Flagging this authentication	3
IS_DEVICE_DERIVED	CHAR	Is the device identified using derived mechanism.	-
NOTES	VARCHAR	Note against this node	255
CACHE	VARCHAR	Cache data for this node log	4000

Database Column Name	Database Column Type	Description	Length
CHALLENGE_CACHE	CLOB	Challenge cache data	-

A.1.3 VCRYPT_TRACKER_NODE

Discover the specifics of the `VCRYPT_TRACKER_NODE` database table.

Description: This table represents a node; device or computer.

Database table name: `VCRYPT_TRACKER_NODE`

Primary Keys: *`NODE_ID`*

Database Column Name	Database Column Type	Description	Length
NODE_ID (PK)	BIGINT	Node Id for this node	16
NODE_VERSION	BIGINT	This keeps track of how many times this node got updated.	16
CREATE_TIME	DATETIME	Date/time for this node.	6
UPDATE_TIME	TIMESTAMP	Last update time for this object	6
RELATED_NODE_ID	BIGINT	Related node	16
RELATION_TYPE	INT	Type of the relation	5
DIG_SIG_COOKIE	VARCHAR	Digital signature cookie	128
SECURE_COOKIE	VARCHAR	Secure cookie	128
REMOTE_IP_ADDR	BIGINT	The IP address from where the client connected	15
REMOTE_HOST	TEXT	The host name from where the client connected	256
FPRINT_ID	BIGINT	Log Id for the fingerprint	16
DIGITAL_FP_ID	BIGINT	Fingerprint Id of the digital cookie request	16
STATUS	INT	Status of this device	3
DEVICE_SCORE	INT	Score for the device for this login	6
IS_DEVICE_DERIVED	CHAR	Is the device identified using derived mechanism.	-
IS_COOKIE_DISABLED	INT	Is the secure cookie disabled for this device or in learn mode.	1
IS_FLASH_DISABLED	INT	Is the flash cookie disabled for this device or in learn mode.	1
NOTES	VARCHAR	Note against this message	255
CACHE	TEXT	Cache	4000

A.1.4 VT_USER_DEVICE_MAP

Discover the specifics of the `VT_USER_DEVICE_MAP` database table.

Description: This table maintains the list of devices the user is using.

Database table name: `VT_USER_DEVICE_MAP`

Primary Key: *`MAP_ID`*

Database Column Name	Database Column Type	Description	Length
MAP_ID (PK)	BIGINT	Map Id	16
USER_ID	BIGINT	Id of the user.	16
NODE_ID	BIGINT	Id of the nodeId.	16
REQUEST_ID	TEXT	Id of the request which last updated this row	256
CREATE_TIME	DATETIME	Date/time when this object was created.	6
UPDATE_TIME	TIMESTAMP	Last update time for this object.	6
LAST_USED_TIME	DATETIME	Last used time for this device.	6
LAST_AUTH_STATUS	INT	Last authentication status for the user using this device.	3
IS_SECURE	CHAR	Is this node secure for this user.	-
TOTAL_COUNT	INT	Total authentication count for this user/ device	10
SUCCESS_COUNT	INT	Total success count for this user/device	10
FAILED_COUNT	INT	Total failed count for this user/device	10
CACHE	TEXT	Cache	4000
IS_COOKIE_DISABLED	INT	Is the secure cookie disabled for this device or in learn mode.	1
IS_FLASH_DISABLED	INT	Is the flash cookie disabled for this device or in learn mode.	1
FPRINT_ID	BIGINT	Fingerprint of secure cookie	16
DIGITAL_FP_ID	BIGINT	Fingerprint Id of the digital cookie request	16

A.1.5 VT_SESSION_ACTION_MAP

Discover the specifics of the `VT_SESSION_ACTION_MAP` database table.

Description: This table maintains the actions for each session.

Database table name: `VT_SESSION_ACTION_MAP`

Primary Key: *`MAP_ID`*

Database Column Name	Database Column Type	Description	Length
MAP_ID (PK)	BIGINT	Map Id	16
CREATE_TIME	DATETIME	Date/time when this object was created.	6
REQUEST_ID	TEXT	Id of the request. This is used to co-relate the post authentication request.	256
TRX_ID	BIGINT	Id for the transaction for this log	16
RUNTIME_TYPE	INT	Type of runtime	6
ACTION	TEXT	Actions for this runtime and session	256
ORIGINAL_ACTION	TEXT	This was the original action, which got overridden finally	256
OVERRIDE_REASON	INT	Override reason	-
ACTION_LIST	TEXT	List of action.	256
SCORE	INT	Score for this runtime and runtime. The same score will appear for all the rows for this transaction	-
IS_FINAL_ACTION	CHAR	Is this final action	-

A.1.6 VT_USER_GROUPS

Discover the specifics of the `VT_USER_GROUPS` database table.

Description: This table contains the user group details.

Database table name: `VT_USER_GROUPS`

Primary Key: *`LOCAL_GROUP_ID`*

Database Column Name	Database Column Type	Description	Length	Enum values
LOCAL_GROUP_ID (PK)	BIGINT	Id for the UserGroup	16	-
EXT_USERGROUP_ID	VARCHAR	External User group Id	255	-
DESCRIPTION	TEXT	Description for this group	2000	-
CREATE_TIME	DATETIME	Date/time creation of this user	6	-
UPDATE_TIME	TIMESTAMP	Date value	6	-
USER_LIST_ID	BIGINT	Id of the userList	16	-

Database Column Name	Database Column Type	Description	Length	Enum values
USERGROUP_STATUS_CODE	INT	Status of the User group	2	• STATUS_ACTIVE • STATUS_DISABLED • STATUS_DELETED
NOTES	TEXT	Note	4000	-

A.1.7 V_FPRINTS

Discover the specifics of the `V_FPRINTS` database table.

Description: This table contains the fingerprints.

Database table name: `V_FPRINTS`

Primary Key: *`FPRINT_ID`*

Database Column Name	Database Column Type	Description	Length
FPRINT_ID (PK)	BIGINT	Id for fingerprint	16
CREATE_TIME	DATETIME	Date/time of this fingerprint	6
FPRINT_TYPE	INT	Type of fingerprinting	6
PATTERN_ID	BIGINT	Id for the pattern this maps to	16
HASH_VALUE	TEXT	Hash value for the fingerprint	512
DATA_VALUE	TEXT	Data value for the fingerprint	4000

A.1.8 V_FP_NV

Discover the specifics of the `V_FP_NV` database table.

Description: This table refers to name value pairs in the fingerprint.

Database table name: `V_FP_NV`

Primary Key *`FP_NV_ID`*

Database Column Name	Database Column Type	Description	Length
FP_NV_ID (PK)	BIGINT	Id for name value	16
FPRINT_ID	BIGINT	Id for the fingerprint	16
ATTR_NAME	VARCHAR	Name of the attribute	64
ATTR_VALUE	TEXT	Value of the attribute	256

A.1.9 V_FP_MAP

Discover the specifics of the `V_FP_MAP` database table.

Description: This table maintains the map for fingerprint.

Database table name: `V_FP_MAP`

Primary Key: *`MAP_ID`*

Database Column Name	Database Column Type	Description	Length
MAP_ID (PK)	BIGINT	Id for map	16
FPRINT_ID	BIGINT	Id for the fingerprint	16
FPRINT_TYPE	INT	Type of fingerprinting	6
ATTR_NAME	VARCHAR	Name of the attribute	64
ATTR_VALUE	TEXT	value of the attribute	256

A.1.10 VCRYPT_COUNTRY

Discover the specifics of the `VCRYPT_COUNTRY` database table.

Description: This table represents the country object.

Database table name: `VCRYPT_COUNTRY`

Primary Key: *`COUNTRY_ID`*

Database Column Name	Database Column Type	Description	Length
COUNTRY_ID (PK)	BIGINT	Id for this country	16
COUNTRY_CODE	VARCHAR	Code of the country	64
COUNTRY_NAME	TEXT	Name of the country	4000
CREATE_TIME	DATETIME	Date/time for this log.	6
UPDATE_TIME	TIMESTAMP	Last update time for this object	6
CONTINENT	VARCHAR	Continent to which this country belongs to	64
NOTES	TEXT	Notes for this country	4000

A.1.11 VCRYPT_STATE

Discover the specifics of the `VCRYPT_STATE` database table.

Description: This table represents state or region of the country.

Database table name: `VCRYPT_STATE`

Primary Key: *`STATE_ID`*

Database Column Name	Database Column Type	Description	Length
STATE_ID (PK)	BIGINT	Id for this state	16
COUNTRY_ID	BIGINT	Id for the country to which this state belongs to	16
STATE_CODE	VARCHAR	Code for the state or region	64
STATE_NAME	TEXT	Name of the state	4000
CREATE_TIME	DATETIME	Date/time for this log	6
UPDATE_TIME	TIMESTAMP	Last update time for this object	6
NOTES	TEXT	Notes for this state	4000

A.1.12 VCRYPT_CITY

Discover the specifics of the `VCRYPT_CITY` database table.

Description: This table represents the city object.

Database table name: `VCRYPT_CITY`

Primary Key: *`CITY_ID`*

Database Column Name	Database Column Type	Description	Length
CITY_ID (PK)	BIGINT	Id for this city	16
STATE_ID	BIGINT	Id for the state to which this city belongs to.	16
CITY_CODE	VARCHAR	Code for the city	64
CITY_NAME	TEXT	Name of the city	4000
CREATE_TIME	DATETIME	Date/time for this log.	6
UPDATE_TIME	TIMESTAMP	Last update time for this object.	6
LATITUDE	VARCHAR	Latitude	20
LONGITUDE	VARCHAR	Longitude	20
TIMEZONE	VARCHAR	Time Zone	20
NOTES	TEXT	Notes for this city	4000

A.1.13 VCRYPT_ISP

Discover the specifics of the `VCRYPT_ISP` database table.

Description: This table represents the ISP listing.

Database table name: `VCRYPT_ISP`

Primary Key: *`ISP_ID`*

Database Column Name	Database Column Type	Description	Length
ISP_ID (PK)	BIGINT	Id for this ISP	16
ISP_NAME	TEXT	Name of the ISP	4000
CREATE_TIME	DATETIME	Date/time for this log	6
UPDATE_TIME	TIMESTAMP	Last update time for this object	6

A.1.14 VCRYPT_IP_LOCATION_MAP

Discover the specifics of the `VCRYPT_IP_LOCATION_MAP` database table.

Description: This table provides the mapping of the IP range to city, state, and country.

Database table name: `VCRYPT_IP_LOCATION_MAP`

Primary Key: *`IP_RANGE_ID`*

Database Column Name	Database Column Type	Description	Length
IP_RANGE_ID (PK)	BIGINT	Id for this range	16
FROM_IP_ADDR	BIGINT	The from IP address	15
TO_IP_ADDR	BIGINT	The to IP address	15
CREATE_TIME	DATETIME	Date/time for this log	-
UPDATE_TIME	TIMESTAMP	Last update time for this object	-
COUNTRY_ID	BIGINT	Id for the country to which this state belongs to	16
STATE_ID	BIGINT	Id for the state to which this IP range belongs to	16
CITY_ID	BIGINT	Id for the city to which this IP range belongs to	16
METRO_ID	BIGINT	Id of the metro for this IP	-
ISP_ID	BIGINT	Id for the ISP to which this IP range belongs to	16
ROUTING_TYPE	INT	IP routing type	3
CONNECTION_TYPE	INT	Connection type	-
CONNECTION_SPEED	INT	Connection speed	-
TOP_LEVEL_DOMAIN	VARCHAR	Top level domain	25
SEC_LEVEL_DOMAIN	VARCHAR	Second level domain	128
ASN	VARCHAR	ASN	25
CARRIER	VARCHAR	Carrier	128
ZIP_CODE	VARCHAR	Zip code	24
DMA	INT	U.S. Designated Market Area, AC Nielsen	6
MSA	INT	Metropolitan Statistical Area	6

Database Column Name	Database Column Type	Description	Length
PMSA	INT	Primary Metropolitan Statistical Area	6
REGION_ID	BIGINT	Id the region	16
PHONE_AREA	VARCHAR	Phone area code	10
IS_SPLIT	CHAR	Is the IP split. If so, in some queries, we might have to do additional checks.	-
COUNTRY_CF	INT	Confidence factor of the country	4
STATE_CF	INT	Confidence factor of the state	4
CITY_CF	INT	Confidence factor of the city	4
NOTES	VARCHAR	Notes for this IP range	255

A.1.15 VT_TRX_DEF

Discover the specifics of the `VT_TRX_DEF` database table.

Description: This table defines the transaction meta data.

Database table name: `VT_TRX_DEF`

Primary Key: *`TRX_DEF_ID`*

Database Column Name	Database Column Type	Description	Length	Enum Values
CREATE_TIME	DATETIME	Date/time creation of this object	6	-
UPDATE_TIME	TIMESTAMP	Date value	6	-
TRX_DEF_ID (PK)	BIGINT	Id for transaction definition	16	-
GLOBAL_ID	VARCHAR	Unique identifier which is used in import and export feature.	255	-
LABEL	TEXT	Name for transaction	4000	-
LABEL_RBKEY	TEXT	Resource bundle key for the name	4000	-
DESCRIPTION	TEXT	Description of the object	4000	-
DESC_RBKEY	TEXT	Resource bundle key for the description	4000	-

Database Column Name	Database Column Type	Description	Length	Enum Values
TRX_DEF_KEY	TEXT	Key name to be used for the transaction, for example bill_pay, etc. This has to be passed in the handleTransactionLog API call. The context map should have an attribute key called transactionType	4000	-
STATUS	INT	Status	2	• STATUS_ACTIVE • STATUS_DISABLED • STATUS_DELETED
NOTES	TEXT	Note for this object	4000	-

A.1.16 VT_TRX_INPUT_DEF

Discover the specifics of the `VT_TRX_INPUT_DEF` database table.

Description: This table contains the definition of transaction input meta data.

Database table name: `VT_TRX_INPUT_DEF`

Primary Key: *`TRX_DEF_ID`*

Database Column Name	Database Column Type	Description	Length	Enum Values
CREATE_TIME	DATETIME	Date/time creation of this object.	6	-
UPDATE_TIME	TIMESTAMP	Date value.	6	-
TRX_DEF_ID (PK)	BIGINT	Id for transaction definition	16	-
GLOBAL_ID	VARCHAR	Unique identifier which is used in import and export feature.	255	-
LABEL	TEXT	Name for transaction.	4000	-
LABEL_RBKEY	TEXT	Resource bundle key for the name	4000	-
DESCRIPTION	TEXT	Description of the object	4000	-

Database Column Name	Database Column Type	Description	Length	Enum Values
DESC_RBKEY	TEXT	Resource bundle key for the description	4000	-
TRX_DEF_KEY	TEXT	Key name to be used for the transaction e.g bill_pay, etc. This has to be passed in the handleTransaction Log API call. The context map should have an attribute key called transactionType	4000	-
STATUS	INT	Status	2	• STATUS_ACTIVE • STATUS_DISABLED • STATUS_DELETED
NOTES	TEXT	Note for this object	4000	-

A.1.17 VT_ENTITY_DEF

Discover the specifics of the `VT_ENTITY_DEF` database table.

Description: This table provides the definition of entity meta data.

Database table name: `VT_ENTITY_DEF`

Primary Key: *`ENTITY_DEF_ID`*

Database Column Name	Database Column Type	Description	Length	Enum Values
CREATE_TIME	DATETIME	Date/time creation of this object	6	-
UPDATE_TIME	TIMESTAMP	Date value	6	-
ENTITY_DEF_ID (PK)	BIGINT	Id for entity definition	16	-
GLOBAL_ID	VARCHAR	Unique identifier which is used in import and export feature	255	-
LABEL	TEXT	Name for entity. For example address, customer	4000	-
LABEL_RBKEY	TEXT	Resource bundle key for the name	4000	-

Database Column Name	Database Column Type	Description	Length	Enum Values
DESCRIPTION	TEXT	Description of the object	4000	-
DESC_RBKEY	TEXT	Resource bundle key for the description	4000	-
ENTITY_DEF_KEY	TEXT	Key of the entity. E.g. address, merchant, etc	256	-
STATUS	INT	Status	2	• STATUS_ACTIVE • STATUS_DISABLED • STATUS_DELETED
KEY_GEN_SCHEME	INT	Key generation scheme. This scheme generates a key which is unique for an entity instance. Points to an enum and supported ones are ByKey, Digest, and so on.	-	-
KEY_GEN_PARAMS	TEXT	Static parameters to be passed to the Java class for key generation	4000	-
NAME_GEN_SCHEME	INT	Name generation scheme. This scheme generates a name which would be how the corresponding entity would be displayed throughout the application in every report. Points to an enum and supported ones are Direct, concatenate, substring, and so on.	-	-

Database Column Name	Database Column Type	Description	Length	Enum Values
NAME_GEN_PARAMS	TEXT	Static parameters to be passed to the Java class for name generation. For example is a delimiter of ','	4000	-
NOTES	TEXT	Note for this object	4000	-

A.1.18 VT_TRX_ENT_DEFS_MAP

Discover the specifics of the `VT_TRX_ENT_DEFS_MAP` database table.

Description: This table defines the association between an entity and the transaction.

Database table name: `VT_TRX_ENT_DEFS_MAP`

Primary Key: *`MAP_ID`*

Database Column Name	Database Column Type	Description	Length	Enum Values
CREATE_TIME	DATETIME	Date/time creation of this object.	6	-
UPDATE_TIME	TIMESTAMP	Date value.	6	-
MAP_ID (PK)	BIGINT	Id for map	16	-
GLOBAL_ID	VARCHAR	Unique identifier which is used in import and export feature.	255	-
LABEL	TEXT	Name for the map.	4000	-
LABEL_RBKEY	TEXT	Resource bundle key for the name	4000	-
DESCRIPTION	TEXT	Description of the object	4000	-
DESC_RBKEY	TEXT	Resource bundle key for the description	4000	-
TRX_DEF_ID	BIGINT	Parent data definition Id	16	-
ENTITY_DEF_ID	BIGINT	Parent data definition Id	16	-
RELATION_TYPE	TEXT	Type of the relation	4000	-
DISP_ORDER	INT	Display order	6	-

Database Column Name	Database Column Type	Description	Length	Enum Values
STATUS	INT	Status	2	• STATUS_ACTIVE • STATUS_DISABLED • STATUS_DELETED
NOTES	TEXT	Note for this object	4000	-

A.1.19 VT_ENT_DEFS_MAP

Discover the specifics of the `VT_ENT_DEFS_MAP` database table.

Description: This table depicts the relationship between an entity and a transaction.

Database table name: `VT_ENT_DEFS_MAP`

Primary Key: *`MAP_ID`*

Database Column Name	Database Column Type	Description	Length
CREATE_TIME	DATETIME	Date/time creation of this object	6
UPDATE_TIME	TIMESTAMP	Date value	6
MAP_ID (PK)	BIGINT	Id for map	16
GLOBAL_ID	VARCHAR	Unique identifier which is used in import and export feature.	255
LABEL	TEXT	Name for the map	4000
LABEL_RBKEY	TEXT	Resource bundle key for the name	4000
DESCRIPTION	TEXT	Description of the object	4000
DESC_RBKEY	TEXT	Resource bundle key for the description	4000
ENTITY_DEF_ID_1	BIGINT	Parent entity definition Id of object 1	16
ENTITY_DEF_ID_2	BIGINT	Parent entity definition Id of object 2	16
RELATION_TYPE	TEXT	Type of the relation	4000
DISP_ORDER	INT	Display order	6
NOTES	TEXT	Note for this object	4000

A.1.20 VT_DATA_DEF

Discover the specifics of the `VT_DATA_DEF` database table.

Description: This table contains the definition of data meta.

Database table name: `VT_DATA_DEF`

Primary Key: *DATA_DEF_ID*

Database Column Name	Database Column Type	Description	Length	Enum Values
CREATE_TIME	DATETIME	Date/time creation of this object.	6	-
UPDATE_TIME	TIMESTAMP	Date value.	6	-
DATA_DEF_ID (PK)	BIGINT	Id for data definition	16	-
GLOBAL_ID	VARCHAR	Unique identifier which is used in import and export feature.	255	-
LABEL	TEXT	Name for data definition.	4000	-
LABEL_RBKEY	TEXT	Resource bundle key for the name	4000	-
DESCRIPTION	TEXT	Description of the object	4000	-
DESC_RBKEY	TEXT	Resource bundle key for the description	4000	-
DATA_DEF_KEY	TEXT	Key of the data. For example, "data", "key", "name", "auto-learning," and so on.	256	-
STATUS	INT	Status	2	• STATUS_ACTIVE • STATUS_DISABLED • STATUS_DELETED
DATA_DEF_TYPE	INT	Type of data definition. Whether it is dynamic or static	5	-
IS_REQUIRED	CHAR	Is this data required by default.	-	-
IS_AUTO_CREATED	CHAR	Whether this auto created.	-	-
NOTES	TEXT	Note for this object	4000	-

A.1.21 VT_DATA_DEF_ELEM

Discover the specifics of the `VT_DATA_DEF_ELEM` database table.

Description: This table provides the definition of elements in data meta.

Database table name: `VT_DATA_DEF_ELEM`

Primary Key: *DATA_DEF_ELEM_ID*

Database Column Name	Database Column Type	Description	Length	Enum Values
CREATE_TIME	DATETIME	Date/time creation of this object	6	-
UPDATE_TIME	TIMESTAMP	Date value	6	-
DATA_DEF_ELEM_ID (PK)	BIGINT	Id for data definition element	16	-
GLOBAL_ID	VARCHAR	Unique identifier which is used in import and export feature	255	-
DEF_KEY	TEXT	Key to identify this data (for example, Transaction.billingAddress.adressLine1, Transaction.amount, and so on). The destination element's keys is different from those of the source element. Within the same data definition, this key has to be unique.	256	-
LABEL	TEXT	Name for column	4000	-
LABEL_RBKEY	TEXT	Resource bundle key for the name	4000	-
DESCRIPTION	TEXT	Description of the object	4000	-
DESC_RBKEY	TEXT	Resource bundle key for the description	4000	-
DATA_DEF_ID	BIGINT	Parent data definition Id (data_def_id from vt_data_def)	16	-
DATA_ROW	INT	Row for this data element	-	-

Database Column Name	Database Column Type	Description	Length	Enum Values
DATA_COL	INT	Column for this data element (starting from 1). This corresponds to the 10 data fields in the VT_TRX_DATA and VT_ENTITY_ONE _PROFILE table for destination elements. For other profile types like "key" and "name", this value determines the sort order for corresponding keygen and namegen scheme.	-	-
IS_ENCRYPTED	CHAR	Is this data element encrypted	-	-
DATA_TYPE	INT	Type of the data (numeric/ alphanumeric types)	-	-
DATA_FORMAT	TEXT	Format of the data (for example, mm/YY for some dates)	4000	-
STATUS	INT	Status	2	• STATUS_ACTIVE • STATUS_DISABLED • STATUS_DELETED
IS_REQUIRED	CHAR	Is this data required by default	-	-
NAME_GEN_SCHEME	INT	Name generation scheme	-	-
NAME_GEN_PARAMS	TEXT	Static parameters to be passed to the Java class for name generation	4000	-
IS_AUTO_CREATED	CHAR	Whether this auto created	-	-
NOTES	TEXT	Note for this object	4000	-

A.1.22 VT_DATA_DEF_MAP

Discover the specifics of the `VT_DATA_DEF_MAP` database table.

Description: This table defines the map between the Objects and the Data Definition.

Database table name: `VT_DATA_DEF_MAP`

Primary Key: *`MAP_ID`*

Database Column Name	Database Column Type	Description	Length
CREATE_TIME	DATETIME	Date/time creation of this object	6
UPDATE_TIME	TIMESTAMP	Date value	6
MAP_ID (PK)	BIGINT	Id for map	16
GLOBAL_ID	VARCHAR	Unique identifier which is used in import and export feature.	255
LABEL	TEXT	Name for the map	4000
LABEL_RBKEY	TEXT	Resource bundle key for the name	4000
DESCRIPTION	TEXT	Description of the object	4000
DESC_RBKEY	TEXT	Resource bundle key for the description	4000
DATA_DEF_ID	BIGINT	Parent data definition Id	16
PARENT_OBJ_TYPE	INT	Type of source object (Points to an enum of types, like 3 for entity, 1 for transaction definition, and so on.)	5
PARENT_OBJECT_ID	BIGINT	Parent to which datadef belongs to (entity_def_id , trx_def_id).	16
RELATION_TYPE	TEXT	Type of the relation ("data", "name," and so on).	4000
NOTES	TEXT	Note for this object	4000

A.1.23 VT_DATA_DEF_TRANS

Discover the specifics of the `VT_DATA_DEF_TRANS` database table.

Description: This table provides the translation from one element to another, for example input transaction to normalized transaction data or transaction to entity.

Database table name: `VT_DATA_DEF_TRANS`

Primary Key: *`ELEM_MAP_ID`*

Database Column Name	Database Column Type	Description	Length	Enum Values
CREATE_TIME	DATETIME	Date/time creation of this object	6	-
UPDATE_TIME	TIMESTAMP	Date value	6	-
ELEM_MAP_ID (PK)	BIGINT	Id for data definition element	16	-

Database Column Name	Database Column Type	Description	Length	Enum Values
GLOBAL_ID	VARCHAR	Unique identifier which is used in import and export feature.	255	-
LABEL	TEXT	Name for this data map	4000	-
LABEL_RBKEY	TEXT	Resource bundle key for the name	4000	-
DESCRIPTION	TEXT	Description of the object	4000	-
DESC_RBKEY	TEXT	Resource bundle key for the description	4000	-
TRANS_SCHEME	INT	Scheme for translation. The value points to an enum of different types of translation schemes.	-	-
TRANS_PARAMS	TEXT	Static parameters to be passed to the Java class for translation	4000	-
SRC_OBJ_TYPE	INT	Type of source object. The value points to an enum for different types of source objects. For example, 3 for entity, 2 for transaction Input, and so on.	5	-
SRC_OBJ_ID	BIGINT	Source object Id (mostly trx_def_id of the corresponding input transaction definition)	16	-
DEST_OBJ_TYPE	INT	Type of destination object. The value points to an enum for different types of destination objects , like 3 for entity, 5 for transaction profile, and so on.	5	-
DEST_OBJ_ID	BIGINT	Destination object Id (map_id from vt_trx_ent_defs_map which denotes the particular relationship type).	16	-

Database Column Name	Database Column Type	Description	Length	Enum Values
RELATION_TYPE	TEXT	Type of the relation	4000	-
STATUS	INT	Status	2	• STATUS_ACTIVE • STATUS_DISABLED • STATUS_DELETED
NOTES	TEXT	Note for this object	4000	-

A.1.24 VT_ELEM_DEF_TRANS

Discover the specifics of the `VT_ELEM_DEF_TRANS` database table.

Description: This table provides the translation from one element to another, for example input transaction to normalized transaction data or transaction to entity.

Database table name: `VT_ELEM_DEF_TRANS`

Primary Key: *`DEST_MAP_ID`*

Database Column Name	Database Column Type	Description	Length	Enum Values
CREATE_TIME	DATETIME	Date/time creation of this object	6	-
UPDATE_TIME	TIMESTAMP	Date value	6	-
DEST_MAP_ID (PK)	BIGINT	Id for data definition element	16	-
GLOBAL_ID	VARCHAR	Unique identifier which is used in import and export feature.	255	-
LABEL	TEXT	Name for this data map	4000	-
LABEL_RBKEY	TEXT	Resource bundle key for the name	4000	-
DESCRIPTION	TEXT	Description of the object	4000	-
DESC_RBKEY	TEXT	Resource bundle key for the description	4000	-
TRANS_SCHEME	INT	Scheme for translation	-	-
TRANS_PARAMS	TEXT	Static parameters to be passed to the Java class for translation	4000	-
TRANS_ID	BIGINT	Translation Id (corresponding elem_map_id from vt_data_def_trans)	16	-

Database Column Name	Database Column Type	Description	Length	Enum Values
DEST_ELEMENT_ID	BIGINT	Destination data element Id (corresponding destination's data_def_elem_id from vt_data_def_elem)	16	-
STATUS	INT	Status	2	• STATUS_ACTIVE • STATUS_DISABLED • STATUS_DELETED
NOTES	TEXT	Note for this object	4000	-

A.1.25 VT_TRANS_SRC_ELEM

Discover the specifics of the `VT_TRANS_SRC_ELEM` database table.

Description: This table contains the source columns for translation.

Database table name: `VT_TRANS_SRC_ELEM`

Primary Key: *`SRC_ELEM_ID`*

Database Column Name	Database Column Type	Description	Length	Enum Values
CREATE_TIME	DATETIME	Date/time creation of this object	6	-
UPDATE_TIME	TIMESTAMP	Date value	6	-
SRC_ELEM_ID (PK)	BIGINT	Id for data definition element	16	-
GLOBAL_ID	VARCHAR	Unique identifier which is used in import and export feature	255	-
LABEL	TEXT	Name for this data map	4000	-
LABEL_RBKEY	TEXT	Resource bundle key for the name	4000	-
DESCRIPTION	TEXT	Description of the object	4000	-
DESC_RBKEY	TEXT	Resource bundle key for the description	4000	-
TRANS_SCHEME	INT	Scheme for translation	-	-

Database Column Name	Database Column Type	Description	Length	Enum Values
TRANS_PARAMS	TEXT	Static parameters to be passed to the Java class for translation	4000	-
DEST_MAP_ID	BIGINT	Destination map Id	16	-
SRC_ELEMENT_ID	BIGINT	Source data element Id	16	-
SORT_ORDER	INT	Row for this data element	-	-
STATUS	INT	Status	2	• STATUS_ACTIVE • STATUS_DISABLED • STATUS_DELETED
NOTES	TEXT	Note for this object	4000	-

A.1.26 VT_TRX_LOGS

Discover the specifics of the `VT_TRX_LOGS` database table.

Description: This table provides the transaction log.

Database table name: `VT_TRX_LOGS`

Primary Key: *`LOG_ID`*

Database Column Name	Database Column Type	Description	Length
LOG_ID (PK)	BIGINT	Log Id	16
CREATE_TIME	DATETIME	Date/time of this transaction.	6
UPDATE_TIME	TIMESTAMP	Last update time for this object.	6
USER_ID	BIGINT	Id of the user.	16
REQUEST_ID	TEXT	Id of the login session.	256
EXT_TRX_ID	VARCHAR	External transaction Id	255
TRX_DEF_ID	BIGINT	Transaction definition Id	16
TRX_TYPE	INT	Transaction type	3
STATUS	INT	Status of the transaction (where applicable)	5
SCORE	INT	Score for this transaction	-
RULE_ACTION	TEXT	Action	256
TRX_FLAG	INT	Flagging this transaction	3
POST_PROCESS_STATUS	INT	Status of the post processing	5

Database Column Name	Database Column Type	Description	Length
POST_PROCESS_RESULT	INT	Status of the post processing	5
TRX_DATA	TEXT	Transaction data as name value pair.	4000
DATA1	TEXT	Data one	256
DATA2	TEXT	Data two	256
DATA3	TEXT	Data three	256
DATA4	TEXT	Data four	256
DATA5	TEXT	Data five	256
DATA6	TEXT	Data six	256
DATA7	TEXT	Data seven	256
DATA8	TEXT	Data eight	256
DATA9	TEXT	Data nine	256
DATA10	TEXT	Data ten	256

A.1.27 VT_TRX_DATA

Discover the specifics of the `VT_TRX_DATA` database table.

Description: This table contains the data associated with the transaction.

Database table name: `VT_TRX_DATA`

Primary Key: *`TRX_DATA_ID`*

Database Column Name	Database Column Type	Description	Length
TRX_DATA_ID (PK)	BIGINT	Transaction data Id	16
TRX_ID	BIGINT	Id of the transaction	16
DATA_DEF_ID	BIGINT	Data definition Id	16
ROW_ORDER	INT	Row order	6
CREATE_TIME	DATETIME	Date/time when this object was created	6
UPDATE_TIME	TIMESTAMP	Last update time for this object	6
DATA1	TEXT	Data one	4000
DATA2	TEXT	Data two	4000
DATA3	TEXT	Data three	4000
DATA4	TEXT	Data four	4000
DATA5	TEXT	Data five	4000
DATA6	TEXT	Data six	4000
DATA7	TEXT	Data seven	4000
DATA8	TEXT	Data eight	4000
DATA9	TEXT	Data nine	4000
DATA10	TEXT	Data ten	4000
NUM_DATA0	BIGINT	Numeric data 0	38

Database Column Name	Database Column Type	Description	Length
NUM_DATA1	BIGINT	Numeric data 1	38
NUM_DATA2	BIGINT	Numeric data 2	38

A.2 Using Geo-Location Data

The OAA/OARM 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 OAA/OARM, IP addresses are stored as long numerals. The following example shows how to 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 example 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))
```

A.3 Building OAA/OARM Custom User Activity Reports

You can build custom user activity reports based on data in the OAA/OARM database schema.

Topics

- Retrieving Entities and Custom User Activities Information
- Discovering Actor or Entity Data Mapping Information
- Discovering Custom User Activity Data Mapping Information

A.3.1 Retrieving Entities and Custom User Activities Information

You can obtain the Custom User Activity Definition key and Entity Definition keys.

Perform the following steps:

1. Log in to the OAA Administration console.
2. In the OAA Administration UI console, click the Application Navigation hamburger menu on the top left.
3. Under **Adaptive Risk Management**, click **Custom Activities**.

   The **Custom Activities Definition Search** page is displayed.
4. Specify criteria in the Search Filter to locate the custom user activity definition you are interested in and press **Enter**.

   The Search Results table displays a summary of the custom user activities definitions that match the search criteria.
5. Click the **Edit** icon in the row for the custom user activity definition you are interested in to view more details.

   The **Edit Custom Activity** page appears.
6. Note down the **Name for this activity**. This is the Custom User Activity Definition Key or the transaction definition key.

   This definition key value is used to map the client/external custom user activity data to custom activity definitions in Oracle Advanced Risk Manager (OARM) server.
   This value is sent while making the API call for creating or updating the custom user activity data in the OARM Server.
7. On the **Custom User Activity Definition Details** or the **Describe Activity** page, click **Next**.

   A list of actors (entities) for the selected custom user activity is displayed.
8. Note down the lists of names in the Actor Name column on the left.
9. Note the **Type** for each of those actors. That is the Actor or Entity Definition Key of the entities.

The definition key is the unique identifier for an actor or entity definition.

A.3.2 Discovering Actor or Entity Data Mapping Information

To discover the actor data mapping information you will need to generate a report.

Perform the following procedures:

- Overview of Data Types
- Discovering Actor or Entity Data Mapping Information
- Building Entity Data SQL Queries and Views

A.3.2.1 Overview of Data Types

Learn about the data types and their descriptions.

The following table lists the data type and their descriptions.

Table A-1 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.

A.3.2.2 Discovering Actor or Entity Data Details

To obtain the actor/entity data detail, such as Data Type, Row, and Column Mappings, you will need to construct your report.

Perform the following steps to generate the report.

1. Log in to the OAA Administration console.
2. In the OAA Administration UI console, click the Application Navigation hamburger menu on the top left.
3. Under **Adaptive Risk Management**, click **Custom Activities**.
   The **Custom Activities Definition Search** page is displayed.
4. Click the **Edit** icon in the row for the custom user activity definition you are interested in to view more details.
   The **Edit Custom Activity** page appears.
5. On the **Custom User Activity Definition Details** or the **Describe Activity** page, click **Next**.
   A list of actors (entities) for the selected custom user activity is displayed.
6. Note the **Type** for each of those actors. That is the Actor or Entity Definition Key of the entities.
   The definition key is the unique identifier for an actor or entity definition.

7. Click the **Edit** icon for the respective actor to view more details.
   The **Edit Actor** page appears. It lists the actor and the data elements contained within it.
8. On the **Edit Actor** page, note down the **Instance Name**, and click **Ok**.
9. Do one of the following to obtain details of how entity data is mapped.
   a. Click the **Map** icon for the respective actor to view more details.
      The **Select or provide Source Data for Actor attributes** page appears. It describes the data items contained within that definition, as well as its source data and mapping information to the model in the OARM server.
   b. Obtain Entity Data mapping 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/Actor Definition Key>
    AND status =1
    )
  )
ORDER BY data_row ASC,
  data_col ASC;
```

A.3.2.3 Building Entity Data SQL Queries and Views

Learn how to create a SQL query and view based on information that reflects the data of a specific actor/entity.

The SQL query in Discovering Actor or Entity Data Mapping Information returns a list of the actor/entity's data fields, together with data type and row and column position. Using this information you will create a SQL query and view that reflects the data of a specific actor/entity.

Note:

`EntityRowN` denotes an entity data row. You would have three `EntityRowN` items, if your entity had three different `data_row` values from the aforementioned SQL query. The aliases must be named `EntityRow1`, `EntityRow2`, and so forth. As illustrated below, you must also take care of the corresponding joins.

```
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>
```

A.3.3 Discovering Custom User Activity Data Mapping Information

To discover custom user activity data mapping information, such as data type, row and column mappings you will need to generate a report.

To obtain the entity data and mapping details using SQL queries, perform the following steps:

Note:

You can also obtain the data mapping information from the OAA Administration console as described in Discovering Actor or Entity Data Details.

1. Use the following SQL query to obtain a list of customer user activities to entity definition mapping IDs.

   ```
   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 obtain details of all custom user activity data fields, together with data type and row and column position.

   ```
   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=<Custom_User_Activity_Key>
           AND status=1
       )
     )
   ORDER BY data_row ASC,
     data_col ASC;
   ```

A.4 Creating Custom Report Example

You can create custom reports on data in the OAA/OARM database schema.

Example 1

This query result will show a list of sessions with `user id`, `login id`, `auth status`, and `location`. You must first create the two date parameters, `fromDate` and `toDate`. The query will look like the following:

```
SELECT s.request_id, s.create_time, s.user_id, s.user_login_id,
country.country_name, statea.state_name, city.city_name
 FROM vcrypt_tracker_usernode_logs s
      INNER JOIN vcrypt_ip_location_map loc ON s.base_ip_addr =
loc.from_ip_addr
      INNER JOIN vcrypt_country country ON loc.country_id =
country.country_id
      INNER JOIN vcrypt_state statea ON loc.state_id = statea.state_id
      INNER JOIN vcrypt_city city ON loc.city_id = city.city_id

WHERE (:fromDate IS NULL OR s.create_time >= :fromDate)
```

```
AND (:toDate IS NULL OR s.create_time <= :toDate)
ORDER BY s.create_time DESC
```

Example 2

Using the OAA/OARM schema, you can generate a custom report for custom user activities. This query result will show a list of custom user activities, request id, status, transaction information for this specific type of transaction, and the creation and modification dates for each key type.

```
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.POST_PROCESS_STATUS,
    trx.POST_PROCESS_RESULT,
    TransactionDataRowN1.NUM_DATA0 NUM_DATA0,
    trx.CREATE_TIME,
    trx.UPDATE_TIME
  FROM VT_TRX_DEF trxDef, VT_TRX_LOGS trx
  LEFT OUTER JOIN VT_TRX_DATA TransactionDataRowN1
  ON (TransactionDataRowN1.TRX_ID = trx.LOG_ID
  AND TransactionDataRowN1.ROW_ORDER = 0)
  WHERE (:fromDate IS NULL OR trx.create_time >= :fromDate)
  AND (:toDate IS NULL OR trx.create_time <= :toDate)
  AND trx.TRX_DEF_ID = trxDef.TRX_DEF_ID and
  trxDef.TRX_DEF_KEY=<Custom_User_Activity_Key>
```

B Understanding OAA/OARM Backup and Recovery

This chapter contains information on backup and recovery techniques. It contains the following sections:

- Backing Up OAA/OARM
- Restoring OAA/OARM

B.1 Backing Up OAA/OARM

Oracle recommends that you periodically take a full backup OAA/OARM data so that you can recover from any unforeseen event and restore your OAA/OARM system.

OAA/OARM consists of file system data, policy and configuration data, and runtime data:

- **File system data** is stored in the NFS volumes. This data includes wallets, the vault, installation properties, and logs.
- **Policy and configuration data** is stored in the database. This data includes assurance levels, rules, policies, actions, groups, customized configuration properties, and transaction definitions.
- **Runtime data** is stored in the database, This data includes user preferences, user sessions, custom user activities, and online transaction and processing data.

A full backup consists of file system data, and a backup of the database.

Oracle also recommends taking policy and configuration data snapshots at various intervals, or when significant policy or configuration changes are made.

Topics:

- Backing Up File System Data
- Backing Up Runtime Data
- Backing Up Policy and Configuration Data

B.1.1 Backing Up File System Data

Oracle recommends that you periodically backup OAA/OARM file system data so that you can recover from any unforeseen event and restore your OAA/OARM system.

OAA/OARM file system data is stored in the NFS volumes; `<NFS_CONFIG_PATH>`, `<NFS_CREDS_PATH>`, `<NFS_LOGS_PATH>`, and `<NFS_VAULT_PATH>`.

You must backup the contents of these NFS volumes, by copying, or creating a compressed zip or tar file, and storing the files in a safe and secure location.

For more information on the NFS volumes, see: Configuring NFS Volumes.

B.1.2 Backing Up Runtime Data

Oracle recommends that you periodically backup OAA/OARM runtime data so that you can recover from any unforeseen event and restore your OAA/OARM system.

To backup runtime data use standard database backup techniques.

For OCI based databases, see Backup Data in Your Databases .

For non OCI based databases, see Backup and Recovery User's Guide.

B.1.3 Backing Up Policy and Configuration Data

Oracle recommends that you periodically backup OAA/OARM policy and configuration data so that you can recover from any unforeseen event and restore your OAA/OARM system.

To backup policy and configuration data:

1. Create a snapshot of the configuration using the `<PolicyUrl>/policy/risk/v1/snapshots` REST API endpoint. For example:

   ```
   curl --location --request POST '<PolicyUrl>/policy/risk/v1/
   snapshots/' \
   --header 'Content-Type: application/json' \
   --header 'Authorization: Basic
   <Base64Encoded(<username>:<password>)>' \
   --data '{
       "name":"Backup Snapshot <DATE>",
       "description": "This is a snapshot from <DATE>"
   }'
   ```

   For details about finding the `PolicyUrl` and authenticating, see OAA Admin API.

   For more details about the snapshot endpoint, see Snapshot REST Endpoints.

   The above command will return a `snapshotId`, for example:

   ```
   {
       "status": "201",
       "message": "Snapshot created successfully.",
       "snapshot": {
           "name": "Backup Snapshot <DATE>",
           "description": "This is a snapshot from <DATE>",
           "snapshotId": "3",
           "createTime": "<DATE>"
       }
   }
   ```

2. Export the snapshot to a zip file using the `snapshotId` returned above, as follows:

   ```
   curl --location --request GET '<PolicyUrl>/policy/risk/v1/snapshots/
   <snapshotId>' \
   ```

```
--header 'Authorization: Basic <Base64Encoded(<username>:<password>)>'
>snapshot<DATE>.zip
```

Store the downloaded zip file in a safe and secure location.

B.2 Restoring OAA/OARM

In order to restore system or runtime data, you must first have created a backup. See Backing Up OAA/OARM.

The steps to restore OAA/OARM depend on the reasons for restoring and whether you are restoring to the same OAA/OARM installation and/or database installation, or to a new installation and/or database installation . The sections below outline the recovery steps based on different scenarios:

- Restoring to an Existing Installation
- Restoring to a New Installation
- Cloning an Installation

B.2.1 Restoring to an Existing Installation

The instructions below can be used to perform the following:

- A full system restore, where you need to perform a full restore of the file system data, the database (runtime data), and policy and configuration data, to the existing installation. This will restore the environment to the point the last full backup was taken.
- A partial restore, where you only need to restore one of either system data, policy and configuration data, the database, or a combination thereof, to the existing installation.

1. If the database needs to be restored, restore the database using standard database recovery techniques. Consult your Oracle Database documentation for further details.
2. If OAA/OARM file system data needs to be restored, follow section **Restoring file system data to an existing installation** in Restoring OAA/OARM File System Data.
3. Restart the OAA/OARM pods by running the following command:

   ```
    kubectl get deployment -n <namespace> | grep <deployment-name> | awk
   '{print $1}' | xargs kubectl rollout restart deployment -n <namespace>
   ```

   For example:

   ```
   kubectl get deployment -n oaans | grep oaainstall | awk '{print $1}' |
   xargs kubectl rollout restart deployment -n oaans
   ```

   The output will look similar to the following:

   ```
   deployment.apps/oaainstall-email restarted
   deployment.apps/oaainstall-fido restarted
   deployment.apps/oaainstall-oaa restarted
   deployment.apps/oaainstall-oaa-admin-ui restarted
   deployment.apps/oaainstall-oaa-kba restarted
   deployment.apps/oaainstall-oaa-policy restarted
   ```

```
deployment.apps/oaainstall-push restarted
deployment.apps/oaainstall-risk restarted
deployment.apps/oaainstall-risk-cc restarted
deployment.apps/oaainstall-sms restarted
deployment.apps/oaainstall-spui restarted
deployment.apps/oaainstall-totp restarted
deployment.apps/oaainstall-yotp restarted
deployment.apps/oaamgmt-oaa-mgmt restarted
```

The above command starts new OAA/OARM pods first, before shutting down the original pods.

Run the following command to check the status of the pods:

```
kubectl get pods -n <namespace>
```

For example:

```
kubectl gets pods -n oaans
```

Once all the previous pods are terminated, and the new pods are at `READY 1/1`, the system is restored:

```
NAME                                 READY   STATUS    RESTARTS   AGE
oaainstall-email-75cccd89f8-9xrgs      1/1     Running
0          5m34s
oaainstall-fido-68777f8cc8-pfw8c       1/1     Running
0          5m34s
oaainstall-oaa-74d5669788-lj5cp        1/1     Running
0          5m34s
oaainstall-oaa-admin-ui-585d55c45b-fzdvk  1/1     Running
0          5m34s
oaainstall-oaa-kba-5b9db9f8db-zwkh2    1/1     Running
0          5m34s
oaainstall-oaa-policy-559fb4d777-qjvwm 1/1     Running
0          5m34s
oaainstall-push-6898c6cb56-l4mg2       1/1     Running
0          5m34s
oaainstall-risk-cc-db558dc5c-qlh8q     1/1     Running
0          5m34s
oaainstall-risk-f48b794bc-j46pz        1/1     Running
0          5m34s
oaainstall-sms-659677b84b-wf7sn        1/1     Running
0          5m34s
oaainstall-spui-6fc8685df9-fhp9w       1/1     Running
0          5m33s
oaainstall-totp-cccd94786-622qd        1/1     Running
0          5m33s
oaainstall-yotp-5fbfd55d4c-d6wqn       1/1     Running
0          5m33s
oaamgmt-oaa-mgmt-94f84ccc6-gwdp2       1/1     Running
0          5m32s
```

4. If you need to import any policy and configuration data from snapshots taken after the last database backup, follow Restoring OAA/OARM Policy and Configuration Data.

B.2.2 Restoring to a New Installation

The instructions below can be used to perform the following:

- A full system restore, where you need to perform a full restore of the file system data, the database (runtime data), and policy and configuration data, to a new installation and database environment. This will restore all file system data, policy and configuration data, and runtime data to the point the last full backup was taken.
- A partial restore where you only need to restore one of either system data, policy and configuration data, the database (runtime data), or a combination thereof, to a new installation.

> **Note:**
>
> If you are only restoring the database to a new database installation, you still need to follow step 3 to restore the OAA/OARM file system data.

> **Note:**
>
> The instructions below assume that if file system data is to be restored to a new installation environment, that the necessary installation prerequsites for that new environment are met. See, Prerequisite Configurations for Installing OAA and OARM

1. If the database needs to be restored to a new environment, restore the database using standard database recovery techniques. Consult your Oracle Database documentation for further details.
2. If you need to restore the file system data to a new installation environment, download the OAA/OARM installation files to that environment. See Downloading Installation Files and Preparing the Management Container.
3. Restore OAA/OARM file system data by following section **Restoring file system data to a new installation** in Restoring OAA/OARM File System Data.
4. If you need to import any policy and configuration data from snapshots taken after the last database backup, follow Restoring OAA/OARM Policy and Configuration Data.

B.2.3 Cloning an Installation

This scenario assumes you want to clone the existing installation to a new environment, using system data, and/or policy and configuration data, from the existing environment. In this scenario no runtime data is restored.

Note:

The instructions below assume that if file system data is to be cloned to a new installation, the necessary installation prerequsites for that new environment are met. See, Prerequisite Configurations for Installing OAA and OARM

1. Download the OAA/OARM installation files to the new system. See Downloading Installation Files and Preparing the Management Container.
2. Restore OAA/OARM file system data to the new system by following section **Restoring file system data to a new installation** in Restoring OAA/OARM File System Data.
3. Restore policy and configuration data by following Restoring OAA/OARM Policy and Configuration Data.

B.2.4 Restoring OAA/OARM File System Data

In order to restore OAA/OARM file system data, you must first have created a backup. See Backing Up File System Data.

Restoring file system data to an existing installation

To restore file system data to the same environment:

1. Copy the file system data from the backup to the NFS volumes `<NFS_CONFIG_PATH>`, `<NFS_CREDS_PATH>`, `<NFS_LOGS_PATH>`, and `<NFS_VAULT_PATH>`.
2. Review the `<NFS_CONFIG_PATH>/installOAA.properties` file and ensure all the external resources such as NFS, the Oracle Database, and OAM OAuth endpoints are available and running.
3. Check if the OAA Management container is running:

   ```
   kubectl get pods -n <namespace> | grep oaamgmt
   ```

   For example:

   ```
   kubectl get pods -n oaans | grep oaamgmt
   ```

4. If the OAA Management container isn't running, you must perform the following steps:
   a. Copy the installOAA.properties from the `<NFS_CONFIG_PATH>` to the `$WORKDIR/oaaimages/oaa-install` directory.
   b. Start the OAA Management by following: Running the Management Container.
5. Continue with the instructions to restart the OAA/OARM pods in section **Restoring to an Existing Installation** .

Restoring file system data to a new installation

To restore file system data to a new environment:

1. Copy the file system data from the backup to the NFS volumes `<NFS_CONFIG_PATH>`, `<NFS_CREDS_PATH>`, `<NFS_LOGS_PATH>`, and `<NFS_VAULT_PATH>`.
2. Review the `<NFS_CONFIG_PATH>/installOAA.properties` file and ensure all the external resources such as NFS, the Oracle Database, and OAM OAuth endpoints are available and running.

   Note:

   If you are restoring to a new system and/or database, make sure all the relevant parameters reference the new system and/or database.

3. Remove the `<NFS_LOGS_PATH>/status.info` file.
4. Copy the installOAA.properties from the `<NFS_CONFIG_PATH>` to the `$WORKDIR/oaaimages/oaa-install` directory.
5. Start the OAA Management Container by following: Running the Management Container.
6. Run the OAA install script from inside the OAA Management container. See Deploying OAA and OARM. This will create a new deployment based on your restored OAA/OARM file system data.

B.2.5 Restoring OAA/OARM Policy and Configuration Data

In order to restore policy and configuration data, you must have either previously created a snapshot, or have the snapshot zip file from a prior backup.

Note:

It is recommended to take a snapshot of the current policy and configuration data before following the steps below. See, Backing Up Policy and Configuration Data.

Restoring from a previous snapshotId

To restore from a previous `snapshotId`:

```
curl --location --request POST '<PolicyUrl>/policy/risk/v1/snapshots/
<snapshotId>/apply' \
--header 'Authorization: Basic <Base64Encoded(<username>:<password>)>' \
--data ''
```

For details about finding the `PolicyUrl` and authenticating, see OAA Admin API.

For more details about the snapshot endpoint, see Snapshot REST Endpoints.

Restoring from a snapshot zip file

To restore from a snapshot zip file:

1. Import the snapshot zip file:

   ```
   curl --location --request POST '<PolicyUrl>/policy/risk/v1/
   snapshots/' \
   --header 'Content-Type: application/octet-stream' \
   --header 'Authorization: Basic
   <Base64Encoded(<username>:<password>)>' \
   --data-binary '@<PATH>/snapshot<DATE>.zip'
   ```

   This will return a `snapshotId`:

   ```
   {
     "status": "201",
     "message": "Snapshot created successfully.",
     "snapshot": {
       "name": "Backup Snapshot <DATE>",
       "description": "This is a snapshot from <DATE>",
       "snapshotId": "4",
       "createTime": "<DATE>"
     }
   }
   ```

2. Apply the snapshot:

   ```
   curl --location --request POST '<PolicyUrl>/policy/risk/v1/
   snapshots/<snapshotId>/apply' \
   --header 'Authorization: Basic
   <Base64Encoded(<username>:<password>)>' \
   --data ''
   ```

   The output will be similar to the following:

   ```
   {
     "serverResponseTime": 1683106368000,
     "clientContext": {
       "invocationContext": {
         "createTime": 1683106335536,
         "invocationId": "d61f7f30-a264-4be0-bb2d-9e5e88c58d19",
         "traceDataXml": "<OARMInvocationContext><invocationId><!
   [CDATA[d61f7f30-a264-4be0-bb2d-9e5e88c58d19]]></
   invocationId><locale></locale><createTime><DATE></createTime></
   OARMInvocationContext>"
       },
       "sessionContext": {
         "sessionId": "",
         "clientId": "",
         "clientVersion": "",
         "userPrincipal": "",
         "ipAddress": "",
         "userAgent": "",
         "createTime": 1683106335537,
         "appName": "UASPolicyApi",
         "accessControlledRole": false,
   ```

```
      "orgAccessList": [],
      "roles": [],
      "traceDataXml": "<OARMSessionContextOARMSessionContext><clientId></
clientId><userAgentString></userAgentString><userPrincipal></
userPrincipal><roles><![CDATA[[]]]></roles><ip></ip><clientVersion></
clientVersion><createTime><DATE></createTime></
OARMSessionContextOARMSessionContext>"
    },
    "taskContext": {
      "taskId": "d61f7f30-a264-4be0-bb2d-9e5e88c58d19",
      "createTime": 1683106335536,
      "traceDataXml": "<OARMTaskContext><taskId><![CDATA[d61f7f30-
a264-4be0-bb2d-9e5e88c58d19]]></taskId><createTime><DATE></createTime></
OARMTaskContext>"
    },
    "traceDataXml":
"<clientContext><OARMSessionContextOARMSessionContext><clientId></
clientId><userAgentString></userAgentString><userPrincipal></
userPrincipal><roles><![CDATA[[]]]></roles><ip></ip><clientVersion></
clientVersion><createTime><DATE></createTime></
OARMSessionContextOARMSessionContext><OARMTaskContext><taskId><!
[CDATA[d61f7f30-a264-4be0-bb2d-9e5e88c58d19]]></
taskId><createTime><DATE></createTime></
OARMTaskContext><OARMInvocationContext><invocationId><![CDATA[d61f7f30-
a264-4be0-bb2d-9e5e88c58d19]]></invocationId><locale></
locale><createTime><DATE></createTime></OARMInvocationContext></
clientContext>"
  },
  "object": true,
  "error": false,
  "success": true,
  "oarmmessages": [],
  "warning": false,
  "serverVersion": "11.1.1.2.0",
  "systemError": false,
  "serverId": "oaainstall-oaa-policy-77bccf774b-48b6s/10.244.1.206",
  "traceDataXml": "<OARMResponse><serverId><![CDATA[oaainstall-oaa-
policy-77bccf774b-48b6s/10.244.1.206]]></serverId><status><!
[CDATA[SUCCESS]]></status><serverResponseTime><DATE></
serverResponseTime><serverVersion><![CDATA[11.1.1.2.0]]></
serverVersion><messageList></messageList></OARMResponse>"
```