Oracle® FMW

Administering Oracle Advanced Authentication and Oracle Adaptive Risk Management

ORACLE®

Contents

Part I   Introduction to Oracle Advanced Authentication, Oracle Adaptive Risk Management, and Oracle Universal Authenticator

1   Introducing Oracle Advanced Authentication

2   Introducing Oracle Adaptive Risk Management

3   Introducing Oracle Universal Authenticator

Part II   Installing Oracle Advanced Authentication, Oracle Adaptive Risk Management, and Oracle Universal Authenticator

4   Supported Architectures

5   Procedure for Installing OAA, OARM, and OUA

Part III   Upgrading OAA, OARM, and OUA

6   Upgrading OAA, OARM, and OUA

7   Rolling Back the Upgrade

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

8   OAAM Features Not Supported or Changed in OARM and OAA

9   Procedure for Transitioning from OAAM to OAA and OARM

10   Viewing the Existing OAAM Policies in the OAA, OARM, and OUA Environment

Part V   Administering Oracle Advanced Authentication

11   Configuring Oracle Advanced Authentication

12   Integrating OAA with Other Products

13   Customizing OAA

14   Understanding Partitioned Schemas

15   Accessibility Features and Tip

Part VI   Managing Oracle Adaptive Risk Management

16   Key Use Cases of OARM for Enhanced Security and Risk Mitigation

17 Device Fingerprinting and Identification

Part VII   Appendices

A Understanding installOAA.properties Parameters

B Advanced Configuration with OAA Override File

C Installing NGINX Ingress Controllers

D Understanding OAA/OARM Schema Reference

E   Understanding OAA/OARM Backup and Recovery

F   Configuring OMA Push Notifications Using Legacy FCM API's

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 installing and configuring Oracle Universal Authenticator (OUA)
- 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:
  - OAA Admin API
  - OAA Runtime API
  - REST API for Risk Service

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.

Part I
Introduction to Oracle Advanced Authentication, Oracle Adaptive Risk Management, and Oracle Universal Authenticator

- Introducing Oracle Advanced Authentication
- Introducing Oracle Adaptive Risk Management
- Introducing Oracle Universal Authenticator

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

- About Oracle Advanced Authentication (OAA)
- Features of Oracle Advanced Authentication (OAA)
- System Components

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 Multi-Factor Authentication (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.

It also can be used with Oracle Universal Authenticator for device authentication with MFA.

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 (Self-Service Portal) 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 Passcode) with Oracle Mobile Authenticator (OMA), Google Authenticator, Microsoft Authenticator, and SafeID/Classic.

    – OTP (One Time Passcode) with email and SMS.

    – Yubikey OTP.

    – FIDO2 - Biometric support using Windows Hello and Mac Touch ID. Support for Yubikey.

    – Knowledge-Based Authentication (KBA).

    – Push Notifications with Oracle Mobile Authenticator.

1.3 System 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.

Self-Service Portal 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 Passcode (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 an integrated system offering comprehensive fraud monitoring, analysis, and tracking based on user location, device, time of day, and other factors, all evaluated against a set of customizable rules.

Topics

- About Oracle Adaptive Risk Management
- Key Features of OARM
- OARM Architecture
- Out-of-the-Box User Authentication Rules Supported
- Distinctive Use Cases of OARM
- Understanding Terminologies in OARM

2.1 About Oracle Adaptive Risk Management

Managing risk is crucial for every organization. Security and risk professionals face the challenge of developing effective models to assess and mitigate the risks posed by each user activity to the organization.

Oracle Adaptive Risk Management (OARM) is an intelligent containerized microservice that helps organizations prevent fraud and misuse across multiple channels of access. It provides intuitive policy management and risk evaluation that is uniquely flexible and effective to prevent fraud and mitigate risks.

OARM can consume external sources of risk data, analyze, and evaluate aggregated risks posed by users and their anomalous behaviors using machine learning capabilities. OARM paired with Oracle Advanced Authentication (OAA) provides actionable risk evaluation to mitigate authentication and business transactional risks. OARM can also be leveraged as a standalone microservice that works as a risk analytics engine that can be integrated with applications or other identity and access management providers to enhance the intelligence of those systems for risk analysis and remedial actions.

2.2 Key Features of OARM

OARM revolves around user activities, which are secured using business friendly rules.

Key differentiating features of OARM include:

- **Risk Evaluation**: Administrators can seamlessly create, edit, and delete security rules in the user-friendly administration console. You can create both access and transaction-based rules from the library of conditions available. The combination of configurable rules, real-time behavioral profiling, and predictive analysis makes OARM a unique offering. It provides a new level of intelligence especially with seeding of data feeds from external sources.

- **Auto-Learning Technology**: OARM leverages a unique mix of real-time and predictive auto-learning technology to profile behaviors and detect anomalies. This enables OARM to recognize high-risk activities and proactively take actions to help prevent fraud and misuse. In addition, the auto-learning capability of OARM allows to constantly evaluate and learn what constitutes typical behavior for both individual users and users as a whole in real time. These techniques enable the system to fully harness both the human talent in your organization and multiple forms of machine learning to combat fraud and misuse.

- **Device Fingerprinting**: OARM offers proprietary, clientless technologies, and an extensible client integration framework for device fingerprinting. Device information is tracked and profiled to detect elevated levels of risk. The fingerprinting process can be run multiple times within a user session, allowing the detection of mid-session changes that may indicate session hijacking.

- **Multi-Factor Challenge Methods**: OARM in conjunction with OAA offers a comprehensive set of modern multi-factor challenge methods, empowering administrators to effortlessly select the challenge mechanisms that best fits with their business needs.

- **Customizable Rules**: OARM is shipped with an out-of-the-box User Authentication activity, which is baked with a rich set of rules that can be readily deployed to enhance business security. The system also provides the flexibility to Administrators to augment the User Authentication activity by adding new rules, removing rules not applicable to business, or adding new user activities for monitoring. Additionally, OARM supports seeding data feeds with certified external sources that can be used in risk analytics. This, when combined with OARM's profiling capabilities, the integration ensures the right mix of seed data for running analytics. The configuration of rules and the management and monitoring of user activities are achieved with an intuitively designed Administration Console. The console allows Administrators to implement rules tailored to their organization without being concerned about the nuances of the underlying system.

2.3 OARM Architecture

OARM with its microservices-based architecture, allows the addition of new capabilities without requiring a costly upgrade process.

OARM integrates with business applications to perform risk analysis and implement remedial actions. Both OARM and OAA are highly extensible due to their microservices-based architecture, enabling seamless integration with REST APIs and the addition of new capabilities without the need for costly upgrades. The intelligent analytics of OARM coupled with the flexible user experience of OAA helps mitigate risk through decision algorithms. When required, modern form factors such as OTP (One Time Passcode), TOTP (Time-based One Time Passcode), YubiKey, and biometric authentication can be leveraged to block, challenge, or allow user activities.

Here's a high-level representation of solution components using OARM and OAA:
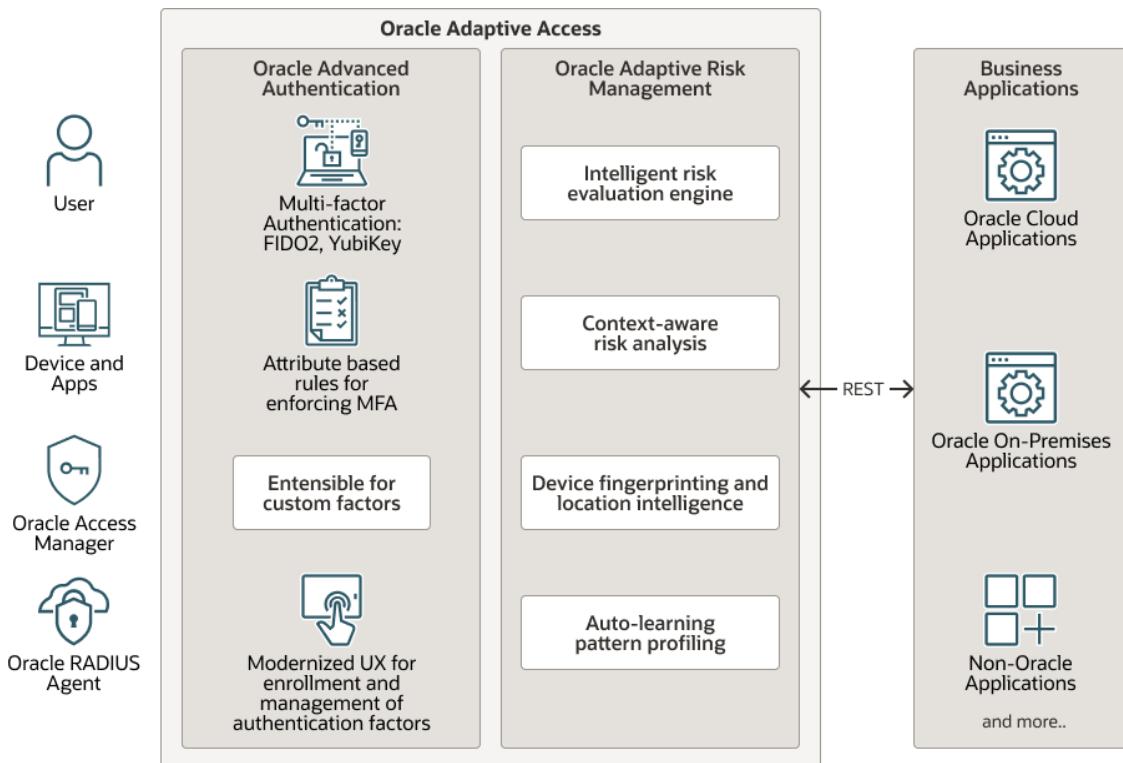
OARM offers flexible deployment options in this release. It can be deployed standalone, or along with OAA, on Kubernetes environments compatible with Cloud Native Computing Foundation (CNCF) Kubernetes in the cloud or on-premises.

2.4 Out-of-the-Box 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.

> **✎ Note:**
>
> To learn how to create a rule effectively, see Add New Rule.

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. To understand how the rule works, see Configuring a Risky IP Use Case. See Configuring a Risky IP Use Case in Oracle Adaptive Risk Management tutorial for detailed instructions on using this rule in OARM. ✎ **Note:** The **Block based on Risky IP** rule was previously referred to as **Challenge based on Risky IP**.
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.

Rule	Description
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.
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. To understand how the rule works, see Configuring a Geo-Velocity Based Use Case. See Configuring a Geo-Velocity Based Use Case in Oracle Adaptive Risk Management tutorial for detailed instructions on using this rule in OARM.
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.

For more details on using rules, see Key Use Cases of OARM for Enhanced Security and Risk Mitigation.

2.5 Distinctive Use Cases of OARM

You can use the OARM User Authentication activity, in conjunction with out-of-the-box rules to perform a wide range of tasks.

The out-of-the-box rules listed in Out-of-the-Box User Authentication Rules Supported allow you to configure use cases such as:

- Risky IP use cases where the Administrator wants to configure IP addresses that are considered as risky for the organization. See Configuring a Risky IP Use Case in Oracle Adaptive Risk Management tutorial for detailed instructions on using this rule in OARM.

- Geo-velocity use cases where the Administrator wants to configure access for authentication of a user based on the distance and the time gap between the users current location and where they last logged in from. See Configuring a Geo-Velocity Based Use Case in Oracle Adaptive Risk Management tutorial for detailed instructions on using this rule in OARM.

In addition to the out-of-the-box User Authentication activity, Administrators can create their own custom activities and create rules using the information collected from it. Rules are customized according to business needs. These rules can be transactional in nature, monitoring various aspects of the user activity that the business is interested in. Some examples of custom activities are internet banking or bill paying in a banking application. You

can add rules that use the information, like the amount involved in the payment, user information, and so on to identify a fraudulent money transfer. See Configuring a Custom Activity Use Case in Oracle Adaptive Risk Management tutorial for detailed instructions on configuring a custom activity in OARM.

For more information on these use cases, see Key Use Cases of OARM for Enhanced Security and Risk Mitigation.

2.6 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.

To view the User Activity rule list, see List User Activity.

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.

See Configuring a Custom Activity Use Case in Oracle Adaptive Risk Management tutorial for detailed instructions on configuring a custom activity in OARM.

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.

To learn how to create a rule effectively, see Add New Rule.

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.

To learn how to create a group effectively, see Create New 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.

To learn how to create a profile effectively, see Create New Profile.

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. To view the **User Sessions** page, see List User Sessions.

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. To monitor a specific user session, see Monitor User Sessions.

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.

3

Introducing Oracle Universal Authenticator

Oracle Universal Authenticator is a unified authentication solution that provides device authentication and cross-platform single-sign on (SSO) to web-based and desktop applications.

Topics

- About Oracle Universal Authenticator

3.1 About Oracle Universal Authenticator

Oracle Universal Authenticator is a unified authentication solution that provides device authentication and cross-platform single-sign on (SSO) to web-based and desktop applications.

Users login to their devices using either passwordless login, or with their Oracle Access Management (OAM) credentials. Users can then access protected applications without the need to enter their single-sign on credentials again.

Oracle Universal Authenticator leverages Oracle Advanced Authentication (OAA) to extend device authentication with multi-factor authentication (MFA), strengthening your organizations security framework, and preventing phishing attacks.

For more information on Oracle Universal Authenticator features and usecases, see Key Oracle Universal Authenticator Features and Use Cases.

Part II

Installing Oracle Advanced Authentication, Oracle Adaptive Risk Management, and Oracle Universal Authenticator

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

- OAA-OARM
- OAA only
- OARM only

Oracle Universal Authenticator (OUA) must be deployed with OAA and OARM, hence the only deployment mode supported for OUA is:

- OAA-OARM-OUA

> **Note:**
>
> OUA requires a client side application to be installed on the device that will be used for device authentication. For more information, see Installing the Oracle Universal Authenticator Client Application.

The procedure to install any of the deployment types is the same, although some prerequisites may differ. The deployment mode is determined by the parameter `common.deployment.mode` defined in the `installOAA.properties`.

> **Note:**
>
> The installation guide in this documentation is based on deploying using the Oracle recommended sandbox architecture. See, Supported Architectures.

4

Supported Architectures

Oracle Advanced Authentication (OAA), Oracle Adaptive Risk Management (OARM), and Oracle Universal Authenticator (OUA) can be deployed in a variety of architectures. For ease of deployment however, Oracle recommends the following based on whether you are deploying a production environment, or a sandbox environment.

Production Deployments

For production deployments, administrators should follow the installation instructions in the Enterprise Deployment Guide for Oracle Identity and Access Management in a Kubernetes Cluster.

The Enterprise Deployment guide provides topology diagrams and step by step instructions on how to build a Kubernetes cluster and deploy OAA (with or without OARM and OUA), either on it's own, or with other Oracle Identity Management products. It also provides automation scripts that simplify the installation experience. The automation scripts can:

- Automate the creation of a Kubernetes cluster on Oracle Cloud Infrastructure (OCI), ready for the deployment of Oracle Identity Management products.

- Automate the deployment of OAA (with or without OARM and OUA) and other Oracle Identity Management products on any compliant Kubernetes cluster.

Sandbox Deployments

For deploying OAA in a sandbox environment, it is assumed you have an existing Oracle Access Management (OAM) environment. The following is a sample OAM environment that can be used for OAA deployment:

Figure 4-1   OAM Sandbox Deployment
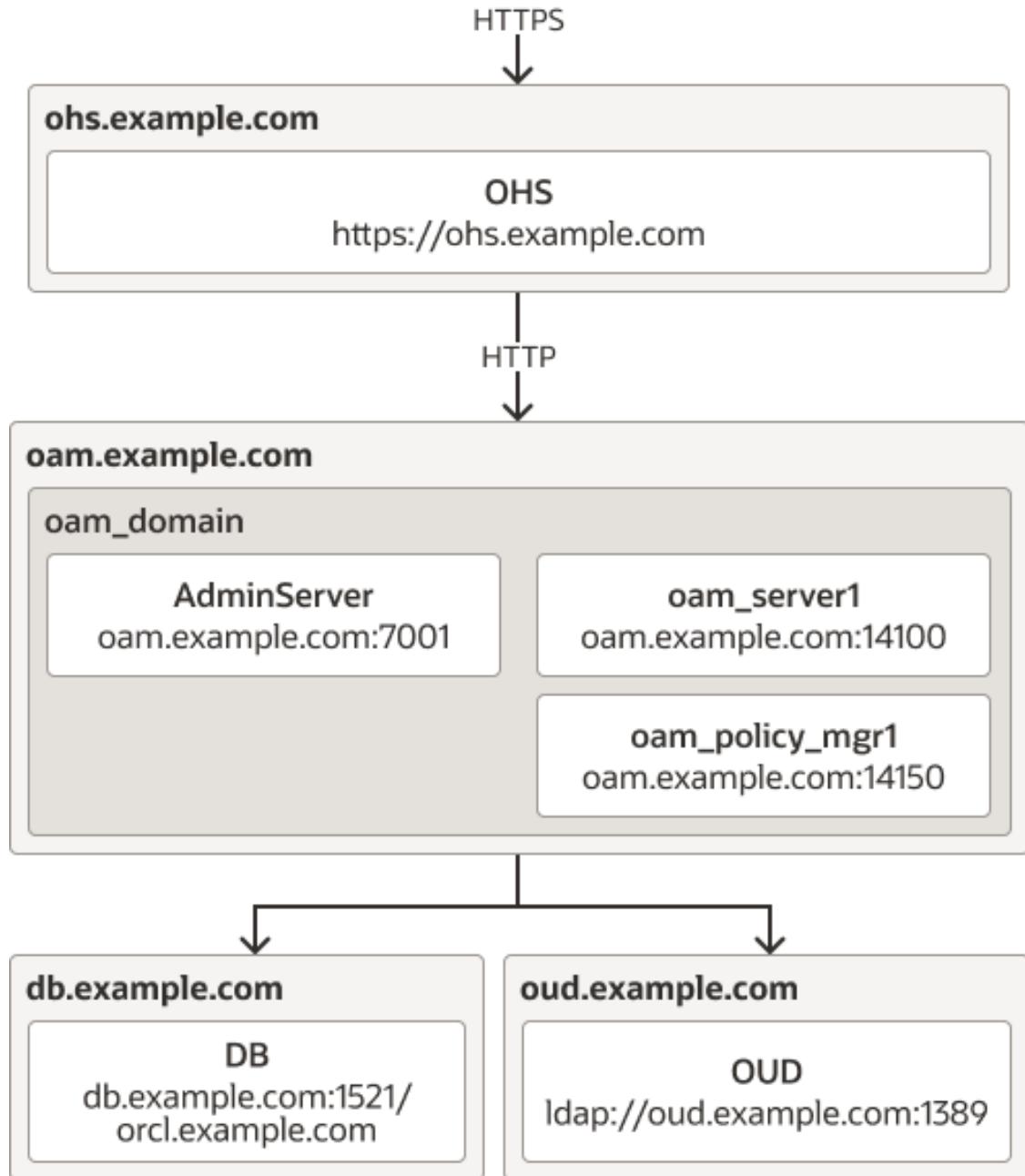
Your OAM environment can be deployed differently to the above, for example you may be terminating SSL at a load balancer in front of OHS, or OUD and the Database are installed on the same server as OAM. However, the following are **mandatory** requirements:

• Oracle HTTP Server (OHS) is deployed on it's own server.

• OHS is used as a proxy to OAM.

• SSL is either terminated at OHS, or at a load balancer in front of OHS.

• Oracle Unified Directory (OUD) is configured with sample users and groups, and extended with OAM Object classes.

• OAM is integrated with OUD and configured to communicate with the OUD LDAP port.

- OAM consoles are protected using an Oracle WebGate and the policies defined in IAMSuiteAgent.

- Both OAM administration and OAM runtime URL's use the same hostname, for example `https://ohs.oracle.com`.

If you do not have an existing OAM environment for use with a sandbox OAA deployment, you can build one following the Configuring Oracle Access Management 12c Sandbox Environment for Oracle Advanced Authentication series.

When OAA, OARM, OUA is deployed the environment will be as follows:

Figure 4-2   OAA and OAM Sandbox Deployment
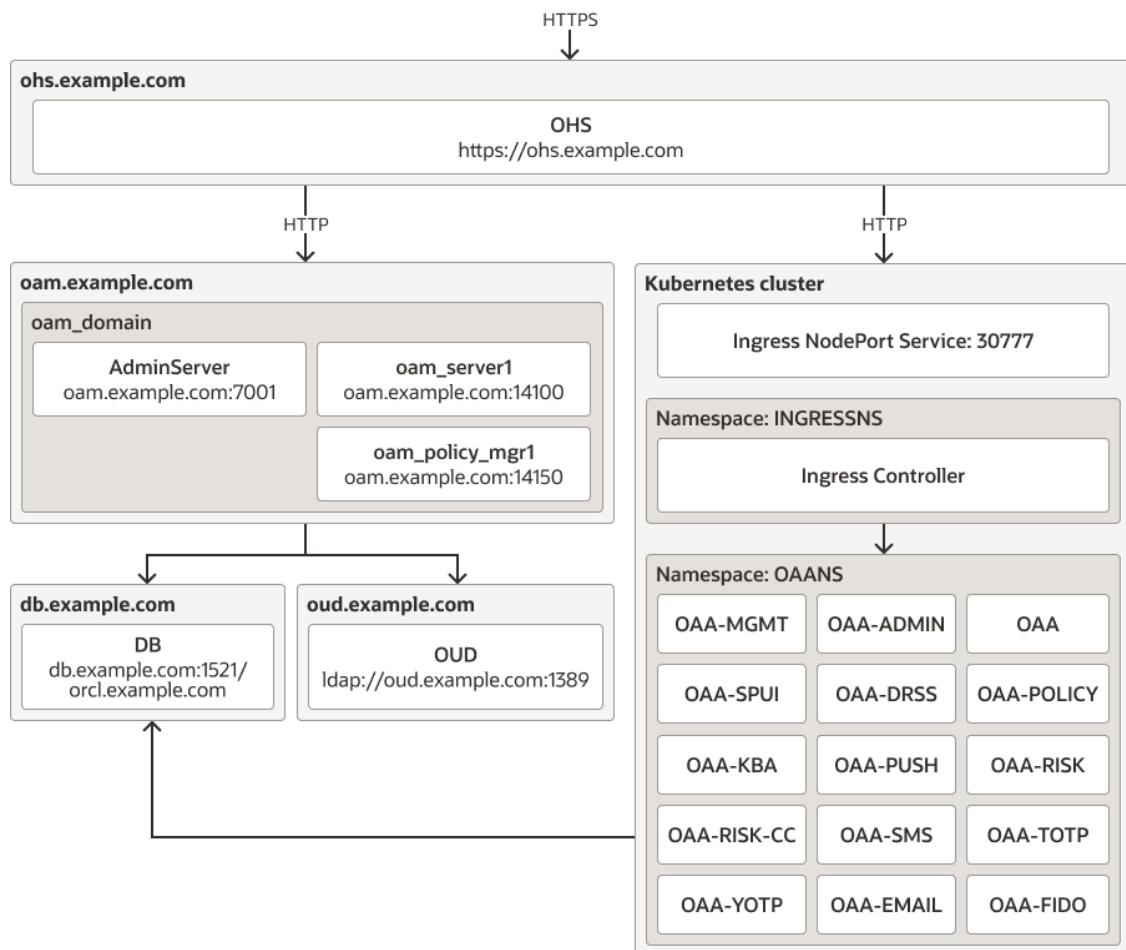
After deployment, all OAA URL's will be accessed via the OHS, for example `https://ohs.example.com`. If using a load balancer in front of OHS then all OAA URL's will be accessed via the load balancer.

> **Note:**
>
> If you intend to use the FIDO2 factor in OAA, administrators should take into consideration that most modern browsers now enforce higher security measures for FIDO2. These browsers will not allow FIDO2 access unless the certificate presented is traceable to a trusted Certificate Authority. Therefore, if you intend to use the FIDO2 factor, the certificate used by OHS (or the load balancer if SSL is terminated there) must be a commerically available certificate, traceable to a trusted Certificate Authority.

Throughout this installation guide, various configuration checkpoints are outlined. These checkpoints take you through basic sanity checks, and gathering variables for the URL's, hostnames, ports, and passwords, that are required later for deployment.

> **Note:**
>
> The installation guide in this documentation is based on deploying on the Oracle recommended sandbox architecture.

5

Procedure for Installing OAA, OARM, and OUA

The following sections show the procedure for installing OAA, OARM and OUA in a sandbox environment:

This section contains the following topics:

- Prerequisite Configurations for Installing OAA, OARM, and OUA
- About the Management Container
- Preparing the Properties file for Installation
- Creating the Management Container
- Deploying OAA, OARM, and OUA
- Printing Deployment Details
- Post Installation Steps for Oracle Universal Authenticator
- Troubleshooting the Installation
- Cleaning Up Installation

Throughout this installation guide, various configuration checkpoints are outlined. These checkpoints take you through basic sanity checks, and gathering variables for the URL's, hostnames, ports, and passwords, that are required later in the `installOAA.properties` file.

> ✏️ **Note:**
>
> The installation guide in this documentation is based on deploying on the Oracle recommended sandbox architecture.

5.1 Prerequisite Configurations for Installing OAA, OARM, and OUA

The OAA, OARM, and OUA installation requires a number of prerequisite configurations. Make sure these prerequisites are met before starting the installation.

The following prerequisites are for all installation types:

- Kubernetes Cluster Requirements
- Installation Host Requirements
- Obtaining the Installation Software
- Configuring the Ingress Controller
- Installing an Oracle Database
- Oracle Access Management Requirements

- Creating Users and Groups in the LDAP Store
- Setting Up a Container Image Registry (CIR)
- Generating Server Certificates and Trusted Certificates
- Validating the Networking Environment
- Creating a Kubernetes Namespace and Secret

5.1.1 Kubernetes Cluster Requirements

OAA, OARM, and OUA is composed of multiple components that run as microservices in 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.

Topics:

- Configuring a Kubernetes Cluster
- Configuring NFS Volumes

5.1.1.1 Configuring a Kubernetes Cluster

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

- The Kubernetes cluster must have a minimum of one master (control plane) node and two worker 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 Supported Virtualization and Partitioning Technologies for Oracle Fusion Middleware.
- 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.

> **✎ Note:**
>
> This documentation does not explain how to configure a Kubernetes cluster given the products can be deployed on any compliant Kubernetes vendor. If you need to understand how to configure a Kubernetes cluster ready for an OAA, OARM, and OUA deployment, you can follow the Enterprise Deployment Guide for Oracle Identity and Access Management in a Kubernetes Cluster.

5.1.1.2 Configuring NFS Volumes

All nodes in the Kubernetes cluster require access to shared volumes on an NFS server. During the 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, PKCS12 files, and the OAA and OUA TAP partner keystores.	*<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>*

5.1.1.3 Configuration Checkpoint

1. Before proceeding make sure you have the following information for your Kubernetes cluster:

Variable	Your Value	Sample Value	Description
<K8S_WORKER_HOST1>,<K8S_WORKER_HOST2>, <K8S_WORKER_HOST3>		`worker1.example.com,worker2.example.com,worker3.example.com`	The fully qualified hostname of the worker nodes.
<NFS_HOST>		`nfs.example.com`	The fuly qualified hostname of the NFS Server used by the Kubernetes Cluster.

Variable	Your Value	Sample Value	Description
<NFS_MOUNT_PATH>		/nfs/mountOAApv	The mount path on NFS server that the Kubernetes worker nodes can access.
<NFS_CONFIG_PATH>		/nfs/mountOAApv/ OAAConfig	The path on NFS server to the *<NFS_CONFIG_PATH>*.
<NFS_CREDS_PATH>		/nfs/mountOAApv/ OAACreds	The path on NFS server to the *<NFS_CREDS_PATH>*.
<NFS_LOGS_PATH>		/nfs/mountOAApv/ OAALogs	The path on NFS server to the *<NFS_LOGS_PATH>*.
<NFS_VAULT_PATH>		/nfs/mountOAApv/ OAAVault	The path on NFS server to the *<NFS_VAULT_PATH>*.

2. Check that Kubernetes is working by running the following command on the bastion node, or master node/control plane:

```
kubectl get nodes
```

Make sure all the nodes return a STATUS of Ready, for example:

```
NAME          STATUS    ROLES          AGE    VERSION
worker-node1  Ready     <none>         76d    v1.29.3+3.el8
master-node   Ready     control-plane  76d    v1.29.3+3.el8
worker-node2  Ready     <none>         76d    v1.29.3+3.el8
worker-node3  Ready     <none>         76d    v1.29.3+3.el8
```

3. From the bastion node, or master node/control plane, check the permissions on the *<NFS_CONFIG_PATH>*, *<NFS_CREDS_PATH>*, *<NFS_LOGS_PATH>*, and *<NFS_VAULT_PATH>*, and make sure they have rwx permissions for all. For example, if the directories are all in *<NFS_MOUNT_PATH>* /nfs/mountOAApv:

```
ls -l /nfs/mountOAApv
drwxrwxrwx. 3 opc opc  3 <DATE> OAAConfig
drwxrwxrwx. 2 opc opc 17 <DATE> OAACreds
drwxrwxrwx. 2 opc opc 34 <DATE> OAALogs
drwxrwxrwx. 2 opc opc  0 <DATE> OAAVault
```

5.1.2 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
```

- You must create a working directory on this installation host. This directory will store files required for installation that will be generated in later sections of this document:

```
mkdir <workdir>
```

    For example:

```
mkdir /workdir
```

5.1.2.1 Configuration Checkpoint

1. Before proceeding make sure you have the following information:

Variable	Your Value	Sample Value	Description
<INSTALL_HOST>		`install.example.com`	Fully qualified hostname of the installation host.
<WORKDIR>		`/workdir`	The working directory created on the installation host.

5.1.3 Obtaining the Installation Software

This section provides steps for downloading the installation software.

All of the following steps should be performed on the *<INSTALL_HOST>*.

1. Download the latest OAA installation Image `<OAA_Image>.zip` from My Oracle Support by referring to the document ID 2723908.1.

2. Create a `<WORKDIR>/oaaimages` directory and copy across the `<OAA_Image>.zip`:

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

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
```

The `installOAA.properties` file will be used later Preparing the Properties file for Installation.

5.1.4 Configuring the Ingress Controller

You must install and configure an Ingress controller on the Kubernetes cluster. As per the sandbox deployment in Supported Architectures, the ingress controller will listen on HTTP.

Run the following commands on the *<INSTALL_HOST>*:

1. Create a namespace for the ingress controller:

```
kubectl create ns ingressns
```

2. Add the helm chart repository for NGINX using the following command:

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

> **Note:**
>
> As the ingress controller is pulled from `https://kubernetes.github.io` you must whitelist this site to pull this image.

3. Update the repository using the following command:

```
helm repo update
```

4. Run the following command on the Kubernetes cluster to install and configure the ingress controller:

```
helm install nginx-ingress -n ingressns --set
controller.service.type=NodePort \
--set controller.service.nodePorts.http=30777 --set
controller.service.nodePorts.https=30443 \
--set controller.config.use-forwarded-headers=true \
--set controller.config.enable-underscores-in-headers=true \
--set controller.admissionWebhooks.enabled=false stable/ingress-nginx
```

> **Note:**
>
> This will start the ingress on HTTP (port 30777) and HTTPS (30443), however the OAA installation will use the HTTP port.

5. To validate that the ingress controller has been successfully created, run the following command:

```
 kubectl get all,ingress -n ingressns
```

The output appears as follows:

```
NAME                                                          READY
STATUS    RESTARTS   AGE
pod/nginx-ingress-ingress-nginx-controller-85985db585-jxcnl   1/1
Running   0          1m30s

NAME                                           TYPE       CLUSTER-
IP     EXTERNAL-IP   PORT(S)                  AGE
service/nginx-ingress-ingress-nginx-controller    NodePort
10.97.137.69   <none>         80:30777/TCP,443:30443/TCP   1m30s

NAME                                             READY   UP-TO-
DATE   AVAILABLE     AGE
deployment.apps/nginx-ingress-ingress-nginx-controller   1/1
1            1               1m30s

NAME
DESIRED   CURRENT   READY   AGE
replicaset.apps/nginx-ingress-ingress-nginx-controller-85985db585
1         1         1       1m30s
```

ORACLE

5.1.5 Installing an Oracle Database

OAA, OARM, and OUA 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, OARM, and OUA supports Oracle Database 12c (12.2.0.1+), 18c, and 19c.

Administrators should be aware of the following:

- For full details on supported database versions, see Oracle Fusion Middleware Supported System Configurations.

- Pluggable databases (PDB) are supported.

- The database parameters should be set as per Repository Creation Utility Requirements.

- 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.

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

5.1.5.1 Configuration Checkpoint

1. Before proceeding make sure you have the following information for the database where the OAA schemas will be loaded:

Variable	Your Value	Sample Value	Description
<DB_HOST>		`db.example.com`	The fully qualified hostname of the database server.
<DB_PORT>		`1521`	The database listener port.
<DB_SERVICE>		`orcl.example.com`	The database service name.
<DB_NAME>		`orcl`	The database name.

Variable	Your Value	Sample Value	Description
<SYS_PWD>		`password`	The password of the SYS user in the database.

2. Make sure the database listener is running:

```
lsnrctl status
```

3. Make sure the database is running and `DB_CREATE_FILE_DEST` is set:

```
SQL>  connect SYS/<password> as SYSDBA;


show parameter DB_CREATE_FILE_DEST;
```

The output should be similar to the following:

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

If there above does not return a value, refer back to Installing an Oracle Database before proceeding.

5.1.6 Oracle Access Management Requirements

The OAA, OARM, and OUA installation needs access to an Oracle Access Management (OAM) deployment.

Topics:

- General OAM Requirements
- Configuring OAuth and Oracle HTTP Server
- Registering OAM TAP Partners

5.1.6.1 General OAM Requirements

This section lists the general requirements for Oracle Access Management.

1. An installation of Oracle Access Management (OAM) 12.2.1.4.0 is required. For Oracle Universal Authenticator (OUA) you must have OAM 12.2.1.4.0 installed with the April 24 Bundle Patch or later. Supported Architectures shows the OAM architecture required.

> **✎ Note:**
>
> If using OUA, and are protecting applications with WebGate(s), it is not supported for those OHS/WebGate(s) to run on the same machine as OAM.

2. The Kubernetes cluster where OAA, OARM, and OUA is to be deployed:

- Must have network connectivity to the OAM deployment.

- Must be time synchronized with the OAM server. It is recommended to use the same time server for both.

- Must be able to resolve the fully qualified hostnames of the OAM server.

3. Take a backup of the existing OAM configuration. See, Performing Backup and Restore and Migration using Snapshot Tool.

5.1.6.2 Configuring OAuth and Oracle HTTP Server

OAA, OARM, and OUA need access to an Oracle Access Management (OAM) deployment with the OAuth service enabled. You must configure Oracle HTTP Server with the relevant entries for OAM OAuth and OAA.

The User Interface (UI) components of OAA, OARM, and OUA (the Administration Console and Self-Service Portal) are protected by OAM OAuth. The OAA installation will configure OAM OAuth for you by:

- Enabling OAuth in OAM.

- Creating the required OAuth components (Identity Domain, Resource, Client) for OAA.

- Creating the required OAM resources within the existing `IAM Suite` application domain.

As per the OAA sandbox deployment architecture in Supported Architectures, OHS is used as a proxy to OAM and OAA. The Oracle HTTP Server (OHS) must be updated with the relevant entries to facilitate the deployment.

Configure Oracle HTTP Server

The instructions below assume that SSL is terminated at OHS, and hence all entries are updated in the relevant SSL `<VirtualHost>` section in the `ssl.conf`. If you have a different architecture, for example SSL is terminated at the load balancer, then update the relevant OHS configuration file for your environment. For more information on updating OHS configutation files, see Working with Oracle HTTP Server.

1. Add the following entries for the RewriteRules for OAuth:

> **Note:**
>
> `OAADomain` will be the OAuth domain that gets created in OAM by the OAA installation.

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

```
domain=OAADomain [PT,QSA,L]
  RewriteRule ^/oauth2/rest/userlogout? /oauth2/rest/userlogout?
domain=OAADomain [PT,QSA,L]
</IfModule>

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

2. Ensure the following OAM entries exist in your OHS configuration file:

> **Note:**
>
> The entries below should already exist given the OAM sandbox deployment
> architecture prerequisite. They are listed here for completeness.

```
#OAM entries
<Location /oam>
  WLSRequest ON
  DynamicServerList OFF
  WLProxySSL ON
  WLProxySSLPassThrough ON
  WLCookieName OAMJSESSIONID
  WebLogicHost <OAM_Managed_Server_Host>
  WebLogicPort <OAM_Managed_Server_Port>
</Location>

<Location /oam/services/rest/auth>
  WLSRequest ON
  DynamicServerList OFF
  WLProxySSL ON
  WLProxySSLPassThrough ON
  WLCookieName OAMJSESSIONID
  WebLogicHost <OAM_Managed_Server_Host>
  WebLogicPort <OAM_Managed_Server_Port>
</Location>

<Location /oam/services/rest/access>
  WLSRequest ON
  DynamicServerList OFF
  WLProxySSL ON
  WLProxySSLPassThrough ON
  WLCookieName OAMJSESSIONID
  WebLogicHost <OAM_Managed_Server_Host>
  WebLogicPort <OAM_Managed_Server_Port>
</Location>

<Location /oamfed>
  WLSRequest ON
  DynamicServerList OFF
  WLProxySSL ON
  WLProxySSLPassThrough ON
```

```
    WLCookieName OAMJSESSIONID
    WebLogicHost <OAM_Managed_Server_Host>
    WebLogicPort <OAM_Managed_Server_Port>
</Location>

# OAM Forgotten Password Page
<Location /otpfp/>
    WLSRequest ON
    DynamicServerList OFF
    WLProxySSL ON
    WLProxySSLPassThrough ON
    WLCookieName OAMJSESSIONID
    WebLogicHost <OAM_Managed_Server_Host>
    WebLogicPort <OAM_Managed_Server_Port>
</Location>

<Location /ms_oauth>
    WLSRequest ON
    DynamicServerList OFF
    WLProxySSL ON
    WLProxySSLPassThrough ON
    WLCookieName OAMJSESSIONID
    WebLogicHost <OAM_Managed_Server_Host>
    WebLogicPort <OAM_Managed_Server_Port>
</Location>

<Location /oauth2>
    WLSRequest ON
    DynamicServerList OFF
    WLProxySSL ON
    WLProxySSLPassThrough ON
    WLCookieName OAMJSESSIONID
    WebLogicHost <OAM_Managed_Server_Host>
    WebLogicPort <OAM_Managed_Server_Port>

</Location>

<Location /.well-known/openid-configuration>
    WLSRequest ON
    DynamicServerList OFF
    PathTrim /.well-known
    PathPrepend /oauth2/rest
    WLProxySSL ON
    WLProxySSLPassThrough ON
    WLCookieName OAMJSESSIONID
    WebLogicHost <OAM_Managed_Server_Host>
    WebLogicPort <OAM_Managed_Server_Port>
</Location>

<Location /.well-known/oidc-configuration>
    WLSRequest ON
    DynamicServerList OFF
    PathTrim /.well-known
    PathPrepend /oauth2/rest
    WLProxySSL ON
    WLProxySSLPassThrough ON
```

ORACLE

```
    WLCookieName OAMJSESSIONID
    WebLogicHost <OAM_Managed_Server_Host>
    WebLogicPort <OAM_Managed_Server_Port>
</Location>

<Location /CustomConsent>
    WLSRequest ON
    DynamicServerList OFF
    WLProxySSL ON
    WLProxySSLPassThrough ON
    WLCookieName OAMJSESSIONID
    WebLogicHost <OAM_Managed_Server_Host>
    WebLogicPort <OAM_Managed_Server_Port>
</Location>

<Location /iam/access>
    WLSRequest ON
    DynamicServerList OFF
    WLProxySSL ON
    WLProxySSLPassThrough ON
    WLCookieName OAMJSESSIONID
    WebLogicHost <OAM_Managed_Server_Host>
    WebLogicPort <OAM_Managed_Server_Port>
</Location>

# WebLogic Console Access
<Location /console>
    WLSRequest ON
    DynamicServerList OFF
    WLProxySSL ON
    WLProxySSLPassThrough ON
    WLCookieName OAMJSESSIONID
    WebLogicHost <OAM_AdminServer_Host>
    WebLogicPort <OAM_AdminServer_Port>
</Location>

<Location /management>
    WLSRequest ON
    DynamicServerList OFF
    WLProxySSL ON
    WLProxySSLPassThrough ON
    WLCookieName OAMJSESSIONID
    WebLogicHost <OAM_AdminServer_Host>
    WebLogicPort <OAM_AdminServer_Port>
</Location>

<Location /consolehelp>
    WLSRequest ON
    DynamicServerList OFF
    WLProxySSL ON
    WLProxySSLPassThrough ON
    WLCookieName OAMJSESSIONID
    WebLogicHost <OAM_AdminServer_Host>
    WebLogicPort <OAM_AdminServer_Port>
</Location>
```

ORACLE

```
<Location /em>
  WLSRequest ON
  DynamicServerList OFF
  WLProxySSL ON
  WLProxySSLPassThrough ON
  WLCookieName OAMJSESSIONID
  WebLogicHost <OAM_AdminServer_Host>
  WebLogicPort <OAM_AdminServer_Port>
</Location>

<Location /oamconsole>
  WLSRequest ON
  DynamicServerList OFF
  WLProxySSL ON
  WLProxySSLPassThrough ON
  WLCookieName OAMJSESSIONID
  WebLogicHost <OAM_AdminServer_Host>
  WebLogicPort <OAM_AdminServer_Port>
</Location>

<Location /access>
  WLSRequest ON
  DynamicServerList OFF
  WLProxySSL ON
  WLProxySSLPassThrough ON
  WLCookieName OAMJSESSIONID
  WebLogicHost <OAM_Policy_Managed_Server_Host>
  WebLogicPort  <OAM_Policy_Managed_Server_Port>
</Location>

<Location /iam/admin>
  WLSRequest ON
  DynamicServerList OFF
  WLProxySSL ON
  WLProxySSLPassThrough ON
  WLCookieName OAMJSESSIONID
  WebLogicHost <OAM_AdminServer_Host>
  WebLogicPort <OAM_AdminServer_Port>
</Location>

<Location /oam/services/rest/11.1.2.0.0>
  WLSRequest ON
  DynamicServerList OFF
  WLProxySSL ON
  WLProxySSLPassThrough ON
  WLCookieName OAMJSESSIONID
  WebLogicHost <OAM_AdminServer_Host>
  WebLogicPort <OAM_AdminServer_Port>
</Location>

<Location /oam/services/rest/ssa>
  WLSRequest ON
  DynamicServerList OFF
  WLProxySSL ON
  WLProxySSLPassThrough ON
  WLCookieName OAMJSESSIONID
```

```
  WebLogicHost <OAM_AdminServer_Host>
  WebLogicPort <OAM_AdminServer_Port>
</Location>

<Location /oam/services>
  WLSRequest ON
  DynamicServerList OFF
  WLProxySSL ON
  WLProxySSLPassThrough ON
  WLCookieName OAMJSESSIONID
  WebLogicHost <OAM_AdminServer_Host>
  WebLogicPort <OAM_AdminServer_Port>
</Location>

<Location /dms>
  WLSRequest ON
  DynamicServerList OFF
  WLProxySSL ON
  WLProxySSLPassThrough ON
  WLCookieName OAMJSESSIONID
  WebLogicHost <OAM_AdminServer_Host>
  WebLogicPort <OAM_AdminServer_Port>
</Location>
```

3. Add the following OAA entries to the OHS configuration file:

> **✎ Note:**
>
> Administrators should be aware of the following:
>
> • The following assumes you will be deploying OAA to a Kubernetes cluster with three worker nodes `<K8S_WORKER_HOST1>`, `<K8S_WORKER_HOST2>`, `<K8S_WORKER_HOST3>`. Modify as required and replace these values with the fully qualified hostnames of the relevant worker nodes as per Configuration Checkpoint.
>
> • Port `30777` is the HTTP port of the ingress controller that you will create later in Configuring the Ingress Controller.

```
## OAA entries
<Location /oaa-admin>
  WLSRequest ON
  WLCookieName OAMJSESSIONID
  DynamicServerList OFF
  WLProxySSL ON
  WLProxySSLPassThrough ON
  WLCookieName OAMJSESSIONID
  WebLogicCluster <K8S_WORKER_HOST1>:30777, <K8S_WORKER_HOST2>:30777,
<K8S_WORKER_HOST3>:30777
</Location>

<Location /admin-ui>
```

```
  WLSRequest ON
  WLCookieName OAMJSESSIONID
  DynamicServerList OFF
  WLProxySSL ON
  WLProxySSLPassThrough ON
  WebLogicCluster
<K8S_WORKER_HOST1>:30777,<K8S_WORKER_HOST2>:30777,<K8S_WORKER_HOST3>:30777
</Location>

<Location /oaa-policy>
  WLSRequest ON
  DynamicServerList OFF
  WLProxySSL ON
  WLProxySSLPassThrough ON
  WLCookieName OAMJSESSIONID
  WebLogicCluster
<K8S_WORKER_HOST1>:30777,<K8S_WORKER_HOST2>:30777,<K8S_WORKER_HOST3>:30777
</Location>

<Location /policy>
  WLSRequest ON
  DynamicServerList OFF
  WLProxySSL ON
  WLProxySSLPassThrough ON
  WLCookieName OAMJSESSIONID
  WebLogicCluster
<K8S_WORKER_HOST1>:30777,<K8S_WORKER_HOST2>:30777,<K8S_WORKER_HOST3>:30777
</Location>

<Location /risk-cc>
  WLSRequest ON
  DynamicServerList OFF
  WLProxySSL ON
  WLProxySSLPassThrough ON
  WebLogicCluster
<K8S_WORKER_HOST1>:30777,<K8S_WORKER_HOST2>:30777,<K8S_WORKER_HOST3>:30777
</Location>

<Location /oua-admin-ui>
  WLSRequest ON
  DynamicServerList OFF
  WLProxySSL ON
  WLProxySSLPassThrough ON
  WLCookieName OAMJSESSIONID
  WebLogicCluster
<K8S_WORKER_HOST1>:30777,<K8S_WORKER_HOST2>:30777,<K8S_WORKER_HOST3>:30777
</Location>

<Location /oaa/runtime>
  WLSRequest ON
  DynamicServerList OFF
  WLProxySSL ON
  WLProxySSLPassThrough ON
  WLCookieName OAMJSESSIONID
  WebLogicCluster
<K8S_WORKER_HOST1>:30777,<K8S_WORKER_HOST2>:30777,<K8S_WORKER_HOST3>:30777
```

ORACLE

```
</Location>

<Location /oaa/rui>
  WLSRequest ON
  DynamicServerList OFF
  WLProxySSL ON
  WLProxySSLPassThrough ON
  WLCookieName OAMJSESSIONID
  WebLogicCluster
<K8S_WORKER_HOST1>:30777,<K8S_WORKER_HOST2>:30777,<K8S_WORKER_HOST3>:30777
</Location>

<Location /oaa/authnui>
  WLSRequest ON
  DynamicServerList OFF
  WLProxySSL ON
  WLProxySSLPassThrough ON
  WebLogicCluster
<K8S_WORKER_HOST1>:30777,<K8S_WORKER_HOST2>:30777,<K8S_WORKER_HOST3>:30777
</Location>

<Location /fido>
  WLSRequest ON
  DynamicServerList OFF
  WLProxySSL ON
  WLProxySSLPassThrough ON
  WLCookieName OAMJSESSIONID
  WebLogicCluster
<K8S_WORKER_HOST1>:30777,<K8S_WORKER_HOST2>:30777,<K8S_WORKER_HOST3>:30777
</Location>

<Location /oaa-email-factor>
  WLSRequest ON
  DynamicServerList OFF
  WLProxySSL ON
  WLProxySSLPassThrough ON
  WLCookieName OAMJSESSIONID
  WebLogicCluster
<K8S_WORKER_HOST1>:30777,<K8S_WORKER_HOST2>:30777,<K8S_WORKER_HOST3>:30777
</Location>

<Location /oaa-sms-factor>
  WLSRequest ON
  DynamicServerList OFF
  WLProxySSL ON
  WLProxySSLPassThrough ON
  WLCookieName OAMJSESSIONID
  WebLogicCluster
<K8S_WORKER_HOST1>:30777,<K8S_WORKER_HOST2>:30777,<K8S_WORKER_HOST3>:30777
</Location>

<Location /oaa-totp-factor>
  WLSRequest ON
  DynamicServerList OFF
  WLProxySSL ON
  WLProxySSLPassThrough ON
```

ORACLE

```
  WLCookieName OAMJSESSIONID
  WebLogicCluster
<K8S_WORKER_HOST1>:30777,<K8S_WORKER_HOST2>:30777,<K8S_WORKER_HOST3>:30777
</Location>

<Location /oaa-push-factor>
  WLSRequest ON
  DynamicServerList OFF
  WLProxySSL ON
  WLProxySSLPassThrough ON
  WLCookieName OAMJSESSIONID
  WebLogicCluster
<K8S_WORKER_HOST1>:30777,<K8S_WORKER_HOST2>:30777,<K8S_WORKER_HOST3>:30777
</Location>

<Location /oaa-yotp-factor>
  WLSRequest ON
  DynamicServerList OFF
  WLProxySSL ON
  WLProxySSLPassThrough ON
  WLCookieName OAMJSESSIONID
  WebLogicCluster
<K8S_WORKER_HOST1>:30777,<K8S_WORKER_HOST2>:30777,<K8S_WORKER_HOST3>:30777
</Location>

<Location /oaa-kba>
  WLSRequest ON
  DynamicServerList OFF
  WLProxySSL ON
  WLProxySSLPassThrough ON
  WLCookieName OAMJSESSIONID
  WebLogicCluster
<K8S_WORKER_HOST1>:30777,<K8S_WORKER_HOST2>:30777,<K8S_WORKER_HOST3>:30777
</Location>

<Location /risk-analyzer>
  WLSRequest ON
  DynamicServerList OFF
  WLProxySSL ON
  WLProxySSLPassThrough ON
  WLCookieName OAMJSESSIONID
  WebLogicCluster
<K8S_WORKER_HOST1>:30777,<K8S_WORKER_HOST2>:30777,<K8S_WORKER_HOST3>:30777
</Location>

<Location /oua>
  WLSRequest ON
  DynamicServerList OFF
  WLProxySSL ON
  WLProxySSLPassThrough ON
  WLCookieName OAMJSESSIONID
  WebLogicCluster
<K8S_WORKER_HOST1>:30777,<K8S_WORKER_HOST2>:30777,<K8S_WORKER_HOST3>:30777
</Location>

<Location /oaa-drss>
```

```
    WLSRequest ON
    DynamicServerList OFF
    WLProxySSL ON
    WLProxySSLPassThrough ON
    WLCookieName OAMJSESSIONID
    WebLogicCluster
<K8S_WORKER_HOST1>:30777,<K8S_WORKER_HOST2>:30777,<K8S_WORKER_HOST3>:30777
</Location>
```

4. Restart Oracle HTTP Server. See, Restarting Oracle HTTP Server Instances .

OAM Resources in OAM

As previously mentioned, the OAA installation will create all the required OAM resources and associated policies in the existing `IAM Suite` application domain.

The following table shows all the resources added by the OAA installation:

Product	Resource Type	Host Identifier	Resource URL	Protection Level	Authentication Policy	Authorization Policy
OAM	HTTP	IAMSuite Agent	`/oauth2/rest/**`	Excluded		
OAM	HTTP	IAMSuite Agent	`/oam/**`	Excluded		
OAM	HTTP	IAMSuite Agent	`/.well-known/ openid- configuration`	Excluded		
OAM	HTTP	IAMSuite Agent	`/iam/access/ binding/api/v10/oap /**`	Excluded		
OAM	HTTP	IAMSuite Agent	`/oam/services/ rest/**`	Excluded		
OAM	HTTP	IAMSuite Agent	`/iam/admin/ config/api/v1/ config/**`	Excluded		
OAM	HTTP	IAMSuite Agent	`/oauth2/rest/ approval`	Protected	OAuth Authentication Policy	Protected Resource Policy
OAM	HTTP	IAMSuite Agent	`/oam/pages/ consent.jsp`	Protected	OAuth Authentication Policy	Protected Resource Policy
OAA	HTTP	IAMSuite Agent	`/oaa-admin/**`	Excluded		
OAA	HTTP	IAMSuite Agent	`/admin-ui/**`	Excluded		
OAA	HTTP	IAMSuite Agent	`/oaa/**`	Excluded		
OAA	HTTP	IAMSuite Agent	`/policy/**`	Excluded		
OAA	HTTP	IAMSuite Agent	`/oaa-email- factor/**`	Excluded		
OAA	HTTP	IAMSuite Agent	`/oaa-sms-factor/**`	Excluded		

ORACLE

Product	Resource Type	Host Identifier	Resource URL	Protection Level	Authentication Policy	Authorization Policy
OAA	HTTP	IAMSuite Agent	`/oaa-totp-factor/**`	Excluded		
OAA	HTTP	IAMSuite Agent	`/oaa-yotp-factor/**`	Excluded		
OAA	HTTP	IAMSuite Agent	`/fido/**`	Excluded		
OAA	HTTP	IAMSuite Agent	`/oaa-kba/**`	Excluded		
OAA	HTTP	IAMSuite Agent	`/oaa-push-factor/**`	Excluded		
OAA	HTTP	IAMSuite Agent	`/risk-analyzer/**`	Excluded		
OAA	HTTP	IAMSuite Agent	`/risk-cc/**`	Excluded		
OAA	HTTP	IAMSuite Agent	`/consolehelp/**`	Excluded		
OAA	HTTP	IAMSuite Agent	`/otpfp/**`	Excluded		
OAA	HTTP	IAMSuite Agent	`/oua/**`	Excluded		
OAA	HTTP	IAMSuite Agent	`/oua-admin-ui/**`	Excluded		
OAA	HTTP	IAMSuite Agent	`/oaa-drss/**`	Excluded		

> **Note:**
>
> If you are using something other than `IAM Suite` and `IAMSuiteAgent` you will need to create all these resources manually.

5.1.6.3 Registering OAM TAP Partners

During installation, two integration agents are created for you in OAA:

- `OAM-OAA-TAP`

- `OAM-OUA-TAP`

In order for the installation to create these agents, you must first create TAP partners in OAM.

Registering OAA as a TAP Partner in OAM

The `OAM-OAA-TAP` OAM integration agent, along with an OAM Authentication Module (`OAA-MFA-Auth-Module`), Authentication Scheme (`OAA-MFA-Scheme`), and Policy (`OAA_MFA-Policy`) are created during installation. These components allow OAM administrators to protect applications with OAM and OAA multi-factor authentication. For example, a user accesses an application protected with the `OAA_MFA-Policy`, and after successful authentication in OAM, is then challenged with a second factor for multi-factor authentication via OAA.

To register `OAM-OAA-TAP` as a TAP partner:

1. On the OAM server, launch a terminal window and enter the following command:

```
cd $OAM_ORACLE_HOME/oracle_common/common/bin
./wlst.sh
```

The output will look similar to the following:

```
Initializing WebLogic Scripting Tool (WLST) ...
Welcome to WebLogic Server Administration Scripting Shell Type help() for
help on available commands
wls:/offline>
```

2. Connect to the OAM Administration Server as follows:

```
connect ('weblogic','<password>','t3://
<OAM_AdminServer_Host>:<OAM_AdminServer_Port>')
```

The output will look similar to the following:

```
Successfully connected to Admin Server "AdminServer" that belongs to
domain "oam_domain".
Warning: An insecure protocol was used to connect to the server. To ensure
on-the-wire security, the SSL port or Admin port should be used instead.
wls:/oam_domain/serverConfig/>
```

3. Run the following command to register the OAA TAP partner:

```
registerThirdPartyTAPPartner(partnerName="OAM-OAA-TAP",
keystoreLocation="<path_to_keystore>", password="<keystore_password>",
tapTokenVersion="v2.0", tapScheme="TAPScheme",
tapRedirectUrl="<redirect_url>")
```

where :

- `<path_to_keystore>` is the location and file name of the keystore to generate.

- `<keystore_password>` is the password to create for the keystore generated.

- `<redirect_url>` is the HTTP or HTTPS URL that you access OAM with at the front end. For example, if you access OAM via `https://ohs.example.com`, then set `tapRedirectUrl` to any URL that is reachable under `https://ohs.example.com`. The URL must be one that returns a 200 OK response when accessed.

For example:

```
registerThirdPartyTAPPartner(partnerName="OAM-OAA-TAP",
keystoreLocation="/tmp/OAMOAAKeyStore.jks", password="password",
tapTokenVersion="v2.0", tapScheme="TAPScheme", tapRedirectUrl="https://
ohs.example.com/oam/pages/login.jsp")
```

The output will look similar to the following:

```
Registration Successful
wls:/oam_domain/serverConfig/>
```

In the example above a keystore `/tmp/OAMOAAKeyStore.jks` will be generated. .

4. Copy the `OAMOAAKeyStore.jks` to the *<WORKDIR>* on the *<INSTALL_HOST>*. See, Installation Host Requirements.

Registering OUA as a TAP Partner in OAM

> **✎ Note:**
>
> If you are performing an installation without OUA, you can ignore this section.

The `OAM-OUA-TAP` agent, along with OAM OUA Policies also created during installation, is used by Oracle Universal Authenticator so users can login to their devices using OAM and a second factor from OAA.

To register `OAM-OUA-TAP` as a TAP partner:

1. In the same WLST session as above, run the following command to register the OUA TAP partner:

    ```
    registerThirdPartyTAPPartner(partnerName="OAM-OUA-TAP",
    keystoreLocation="<path_to_keystore>", password="<keystore_password>",
    tapTokenVersion="v2.0", tapScheme="TAPScheme",
    tapRedirectUrl="<redirect_url>")
    ```

    where :

    *   `<path_to_keystore>` is the location and file name of the keystore to generate.

    *   `<keystore_password>` is the password to create for the keystore generated.

    *   `<redirect_url>` is the HTTP or HTTPS URL that you access OAM with at the front end. For example, if you access OAM via `https://ohs.example.com`, then set `tapRedirectUrl` to any URL that is reachable under `https://ohs.example.com`. The URL must be one that returns a 200 OK response when accessed.

    For example:

    ```
    registerThirdPartyTAPPartner(partnerName="OAM-OUA-TAP",
    keystoreLocation="/tmp/OAMOUAKeyStore.jks", password="password",
    tapTokenVersion="v2.0", tapScheme="TAPScheme", tapRedirectUrl="https://
    ohs.example.com/oam/pages/login.jsp")
    ```

    The output will look similar to the following:

    ```
    Registration Successful
    wls:/oam_domain/serverConfig/>
    ```

    In the example above a keystore `/tmp/OAMOUAKeyStore.jks` will be generated.

2. Copy the `OAMOUAKeyStore.jks` to the *<WORKDIR>* on the *<INSTALL_HOST>*. See, Installation Host Requirements.

3. Run the following command to exit wlst:

```
exit()
```

5.1.6.4 Configuration Checkpoint

OHS Variables

1. Before proceeding make sure you have the following information:

Variable	Your Value	Sample Value	Description
<WEB_HOST>		`https:// ohs.oracle.com`	The URL of the OHS server that is used as the entry point to OAM. If you are using a load balancer in front of OHS, also collect the URL of the *<LBR_HOST>*.

Variable	Your Value	Sample Value	Description
<OAM_ADMIN_USER>		`oamadmin`	The username of the OAM administration user who logs into the OAM Administration console (`/oamconsole`).
<OAM_ADMIN_PASSWORD>		`password`	The password for the OAM administration user.
<OAM_ADMIN_BASE64>		`b2FtYWRtaW46cGFzc3 dvcmQ=`	The BASE64 encoded value of *<OAM_ADMIN_USER>:<OAM_ADMIN_PASSWORD>*. To find the BASE64 encoded version run: `echo -n <OAM_ADMIN_USER>: <OAM_ADMIN_PASSWO RD>	base64`
<IDSTORE>		`OUDStore`	The Default User Identity Store used by OAM. This value can be found by logging into the OAM Administration Console and navigating to **Configuration** > **User Identity Stores** > **Default Store**.

ORACLE

Variable	Your Value	Sample Value	Description
<OUA_TAPFILE_LOCATION>		`/workdir/ OAMOUAKeyStore.jks`	The location of the `<WORKDIR>/ OAMOUAKeyStore.jks` on the *<INSTALL_HOST>*.
<OUA_TAPFILE_PASSWORD>		`cGFzc3dvcmQ=`	The BASE64 encoded password of the `OAMOUAKeyStore.jks`. To find the BASE64 encoded version run: `echo -n <password> | base64`
<OAA_TAPFILE_LOCATION>		`/workdir/ OAMOAAKeyStore.jks`	The location of the `<WORKDIR>/ OAMOAAKeyStore.jks` on the *<INSTALL_HOST>*.
<OAA_TAPFILE_PASSWORD>		`cGFzc3dvcmQ=`	The BASE64 encoded password of the `OAMOAAKeyStore.jks`. To find the BASE64 encoded version run: `echo -n <password> | base64`

5.1.7 Creating Users and Groups in the LDAP Store

Oracle Advanced Authentication (OAA) requires two groups to be configured in the LDAP store used by Oracle Access Management (OAM):

- **OAA-Admin-Role**, which is used to authenticate administrator users who are permitted to access the Administration Console.
- **OAA-App-User**, which contains the list of users who are permitted to access the Self-Service Portal.

These groups, along with the OAA Administration user `oaaadmin`, are created in your LDAP store during the OAA installation (unless they already exist).

> **Note:**
>
> As described in Supported Architectures, the LDAP store used by OAM must be extended with OAM Object classes. For more information, see Using Password Policy.

The installation can optionally add all your existing LDAP users (in the defined user search base) to the **OAA-App-User** group, and add the `obpsftid: true` attribute for each user.

> **Note:**
>
> The `obpsftid: true` is a requirement for persistent login and Oracle Universal Authenticator.

If you do not want to add all of your defined LDAP user search base to the **OAA-App-User** group, then you must add the users manually, post-installation of OAA. Similarly, if using OUA you must also add the LDAP attribute `obpsftid: true` to each user manually.

5.1.7.1 Configuration Checkpoint

1. Before proceeding make sure you have the following information:

Variable	Your Value	Sample Value	Description
<LDAP_HOST>		`oud.example.com`	The fully qualified hostname of the LDAP server.
<LDAP_SERVER>		`ldap:// oud.example.com:13 89`	The LDAP server protocol, hostname and port.
<LDAP_ADMIN_USER >		`cn=oudadmin`	The user name of the directory administrator.
<LDAP_ADMIN_PWD>		`password`	The password of the directory administrator.
<LDAP_USER_SEARC HBASE>		`cn=Users,dc=exampl e,dc=com`	The location in the directory where names of users are stored.
<LDAP_GROUP_SEA RCHBASE>		`cn=Groups,dc=examp le,dc=com`	The location in the directory where groups/ roles are stored.

5.1.8 Setting Up a Container Image Registry (CIR)

During the management container installation, container images are pushed to a Container Image Registry (CIR). During deployment, images are pulled from the same registry. You must therefore setup a Container Image Registry as a prerequisite. This registry must be accessible from all nodes in the Kubernetes cluster where OAA, OARM, and OUA 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. 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

- oaa-drss

Additional Images

During installation the following additional image is installed:

- `oraclelinux:8-slim` from `https://ghcr.io/oracle`

Administrators must whitelist this sites to allow the Kubernetes cluster to pull this image.

If you cannot whitelist this site, then you must pull the image down manually and store them in your container registry. For example to pull the `ghcr.io/oracle` image:

```
podman pull ghcr.io/oracle/oraclelinux:8-slim
```

In order for the installation to know about the location of the `oraclelinux:8-slim` image, you must edit the `installOAA.properties` and in the **## 5. Chart configuration#** section, set `install.global.testrepo` to the location of your container registry.

> **Note:**
>
> This parameter is not shown in the `installOAA.properties` file by default.

See, Preparing the Properties file for Installation.

5.1.8.1 Configuration Checkpoint

1. Before proceeding make sure you have the following information:

Variable	Your Value	Sample Value	Description
<CIR_HOST>		`cir.example.com`	The fully qualified hostname of the Container Image Registry

Variable	Your Value	Sample Value	Description
<CIR_REPOSITORY>		`cir.example.com/ repository/oaa`	The repository where the OAA images will be pushed to.

5.1.9 Generating Server Certificates and Trusted Certificates

OAA uses SSL for communication. For production environments it is recommended to use a commercially available certificate, traceable to a trusted Certificate Authority. For OAA sandbox environments, the OAA installation can generate self-signed certificates which have a validity period of 6 months.

If you want to use self-signed certificates you can skip this section.

If you want to use a commercial certificate then follow:

- Using a Third Party CA for Generating Certificates

> **✎ Note:**
>
> Administrators should be aware that certificates have expiry dates. The expiry date of commercial certificates vary, and self-signed certificates expire after 6 months. It is recommend to renew the certificates about one month before expiry. For more information, see Certificate Management and Expiry.

5.1.9.1 Using a Third Party CA for Generating Certificates

The following steps show how to use a third party Certificate Authority (CA) for generating your certificates:

1. On the *<INSTALL_HOST>* create an `oaassl` directory in *<WORKDIR>* and navigate to the folder, for example:

   ```
   mkdir /workdir/oaassl
   cd /workdir/oaassl
   ```

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 CA file(s). If your CA does not have a certificate chain, replace `bundle.pem` with `rootca.pem`:

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

When prompted enter and verify the Export Password.

> **✎ Note:**
>
> Setting an export password is mandatory.

9. Create a Server Certificate PKCS12 file (`cert.p12`). If your CA does not have a certificate chain, replace `bundle.pem` with `rootca.pem` in the following command :

```
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:**
>
> Setting an export password is mandatory.

10. Copy the `cert.p12` and `trust.p12` to the *<WORKDIR>*, for example:

```
cp /workdir/oaassl/*.p12 /workdir/
```

> **Note:**
>
> In releases prior to December 24 you also had to import the OAM certificates to the `trust.p12`. From December 24 onwards, this action is no longer required as the installation will download the correct certificates from OAM and import them for you.

5.1.9.2 Configuration Checkpoint

1. Before proceeding make sure you have the following information:

> **Note:**
>
> If you intend to use self-signed certificates created by the installation, you can ignore this section.

Variable	Your Value	Sample Value	Description
<USER_CERT_P12>		`/workdir/cert.p12`	The location of the *<WORKDIR>*/`cert.p12` on the *<INSTALL_HOST>*. This is only required if you generated third party certificates.
<USER_CERT_P12_PWD>		`password`	The password for the `cert.p12`. This is only required if you generated third party certificates.

Variable	Your Value	Sample Value	Description
<TRUST_CERT_P12>		`/workdir/trust.p12`	The location of the *<WORKDIR>/* `trust.p12` on the *<INSTALL_HOST>*. This is only required if you generated third party certificates.
<TRUST_CERT_P12_ PWD>		`password`	The password for the `trust.p12`. This is only required if you generated third party certificates.

5.1.10 Validating the Networking Environment

Perform the checks described in this section to ensure that your environment is ready for a deployment. If any of the checks fail, then you must resolve before proceeding.

> **Note:**
>
> The variables used in this section are based on the values collected in the earlier Configuration Checkpoint sections

Bastion or Master/Control Plane

1. From the Kubernetes bastion, or master/control plane node, run the following:

   - `nc -zv <WEB_HOST> <PORT>`

     > **Note:**
     >
     > In this case *<WEB_HOST>* is the fully qualified hostname of the OHS (or load balancer if one is used in front of OHS), and `<PORT>` is the configured SSL port.

   - `nc -zv <DB_HOST> <DB_PORT>`

   - `nc -zv <LDAP_HOST> <LDAP_PORT>`

   For all of the above you should receive the following:

   ```
   Ncat: Connected to <IP_ADDRESS>:<PORT>.
   Ncat: 0 bytes sent, 0 bytes received in 0.01 seconds.
   ```

2. Most containers do not have built in networking tools to allow you to check that DNS resolution is working correctly. The easiest way to validate the changes is to use a lightweight container with the network tools installed, such as alpine:

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

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

> **Note:**
>
> The above assumes your Kubernetes cluster has access to the internet to access `docker.io`.

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

b. Inside the container you can then run `nslookup` against the *<DB_HOST>*, *<LDAP_HOST>*, *<WEB_HOST>* (and *<LBR_HOST>* if using a load balancer), and *<CIR_HOST>*. For example:

```
nslookup ohs.example.com
```

Make sure the hostnames resolve correctly. If you have problems resolving any of the hostnames, contact the Kubernetes administrator to resolve before proceeding.

Web Tier (OHS)

From the *<WEB_HOST>*, run the following to check you can connect to the ingress controller port:

- `nc -zv <K8S_WORKER_HOST1> 30777`

- `nc -zv <K8S_WORKER_HOST2> 30777`

- `nc -zv <K8S_WORKER_HOST3> 30777`

For all of the above you should receive the following: :

```
Ncat: Connected to <IP_ADDRESS>:30777.
Ncat: 0 bytes sent, 0 bytes received in 0.01 seconds.
```

OAA Installation Host

From the *<INSTALL_HOST>*, run the following:

- `ping <CIR_HOST>`

- `kubectl get nodes`

- `curl -k -I https://container-registry.oracle.com`

> **Note:**
>
> You should receive `HTTP/1.0 200 Connection Established`

5.1.11 Creating a Kubernetes Namespace and Secret

Create a Kubernetes namespace and secret for the deployment.

1. Run the following command on the *<INSTALL_HOST>* to create a Kubernetes namespace `oaans` for the deployment:

```
kubectl create namespace oaans
```

2. Create a Kubernetes secret called `dockersecret` 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/OUA 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
```

5.2 About the Management Container

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

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

5.2.1 Components of the Management Container

This section provides an overview of important files and folders in the management container pod.

Table 5-1   Management Container Files and Folder Reference

Files and Folders	Description
OAA.sh	This script file is used to install OAA, OARM, and OUA. The `installOAA.properties` file must be given as an argument to the script for installing OAA, OAA-OARM, OARM, and OAA-OARM-OUA.
installsettings	This folder contains the `oaaoverride.yaml` that can be customized to set the `replicaCount` for some of the services in OAA, OARM, and OUA. 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, OARM, and OUA services.
libs	This folder contains the following files: • `OAAAuthnPlugin.jar`: this plugin is used for integrating OAM with OAA. • `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, OARM, and OUA 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 that get copied, created, and used during installation: • `trust.p12` • `cert.p12` • `k8sconfig` • `helmconfig` • `OAMOUAKeyStore.jks` • `OAMOAAKeyStore.jks`
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, OARM, and OUA deployment. It stores the file based vault (if not using OCI based vault).

5.2.2 Preset Environment Variables in Management Container

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

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.

Environment Variable	Description
CREDS_DIR	This is a NFS volume *<NFS_CREDS_PATH>* used to store credentials, such as helmconfig, kubeconfig, tap partner keystores, 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.
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.

5.2.3 Mounted Volumes in the Management Container

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

Mounted Volumes in Management Container

The information in this section releates to the NFS volumes you created in Configuring NFS Volumes.

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>*.	**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>*.	**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 Kubernetes and Helm configuration files, SSH keys, PKCS12 files, and the OAA and OUA TAP partner keystores. This maps to NFS volume *<NFS_CREDS_PATH>*.	**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>*	**Read-Write-Execute** The NFS volume *<NFS_VAULT_PATH>* must have Read-Write-Execute permissions for all.

5.3 Preparing the Properties file for Installation

You can customize the OAA, OARM, and OUA installation by setting properties in the `installOAA.properties` file. The `installOAA.properties` is used by the Management Container installation script and is copied to the *<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, OARM, and OUA.

The following sections show how to prepare the `installOAA.properties` file based on deploying using the Oracle recommended sandbox architecture in Supported Architectures:

- Gathering Variables
- Editing the installOAA.properties

5.3.1 Gathering Variables

List of Variables

Throughout the Prerequisite Configurations for Installing OAA, OARM, and OUA chapter, you should have gathered variables during the **Configuration Checkpoints** sections. These variables will be used to populate the parameters in the `installOAA.properties`. The variables required for the `installOAA.properties` are shown below for completeness:

> **Note:**
>
> The variables are listed in the order they were collected in the **Configuration Checkpoints** sections.

Variable	Your Value	Sample Value	Description
<NFS_HOST>		`nfs.example.com`	The fully qualified hostname of the NFS Server used by the Kubernetes Cluster.
<NFS_CONFIG_PATH>		`/nfs/ mountOAApv/ OAAConfig`	The path on NFS server to the *<NFS_CONFIG_PATH>*
<NFS_CREDS_PATH>		`/nfs/ mountOAApv/ OAACreds`	The path on NFS server to the *<NFS_CREDS_PATH>*
<NFS_LOGS_PATH>		`/nfs/ mountOAApv/ OAALogs`	The path on NFS server to the *<NFS_LOGS_PATH>*
<NFS_VAULT_PATH>		`/nfs/ mountOAApv/ OAAVault`	The path on NFS server to the *<NFS_VAULT_PATH>*
<INSTALL_HOST>		`install.example .com`	Fully qualified hostname of the installation host.
<WORKDIR>		`/workdir`	The working directory created on the installation host.

Variable	Your Value	Sample Value	Description
<DB_HOST>		`db.example.com`	The fully qualified hostname of the database server.
<DB_PORT>		`1521`	The database listener port.
<DB_SERVICE>		`orcl.example.com`	The database service name.
<DB_NAME>		`orcl`	The database name.
<SYS_PWD>		`password`	The password of the SYS user in the database.
<WEB_HOST>		`https://ohs.oracle.com`	The fully qualified hostname of the OHS server that is used as the entry point to OAM. If you are using a load balancer in front of OHS, also collect the fully qualified hostname of the load balancer *<LBR_HOST>*.
<OAM_ADMIN_USER>		`oamadmin`	The username of the OAM administration user who logs into the OAM Administration console (`/oamconsole`).
<OAM_ADMIN_PASSWORD>		`password`	The password for the OAM administration user.
<OAM_ADMIN_BASE64>		`b2FtYWRtaW46cGFzc3dvcmQ=`	The BASE64 encoded value of *<OAM_ADMIN_USER>:<OAM_ADMIN_PASSWORD>*
<IDSTORE>		`OUDStore`	The Default User Identity Store used by OAM.
<OUA_TAPFILE_LOCATION>		`/workdir/OAMOUAKeyStore.jks`	The location of the `<WORKDIR>/OAMOUAKeyStore.jks` on the *<INSTALL_HOST>*.
<OUA_TAPFILE_PASSWORD>		`cGFzc3dvcmQ=`	The BASE64 encoded password of the `OAMOUAKeyStore.jks`.
<OAA_TAPFILE_LOCATION>		`/workdir/OAMOAAKeyStore.jks`	The location of the `<WORKDIR>/OAMOAAKeyStore.jks` on the *<INSTALL_HOST>*.
<OAA_TAPFILE_PASSWORD>		`cGFzc3dvcmQ=`	The BASE64 encoded password of the `OAMOAAKeyStore.jks`.
<LDAP_SERVER>		`ldap://oud.example.com:1389`	The LDAP server protocol, hostname and port.
<LDAP_ADMIN_USER>		`cn=oudadmin`	The user name of the directory administrator.
<LDAP_ADMIN_PWD>		`password`	The password of the directory administrator.
<LDAP_USER_SEARCHBASE>		`cn=Users,dc=example,dc=com`	The location in the directory where names of users are stored.
<LDAP_GROUP_SEARCHBASE>		`cn=Groups,dc=example,dc=com`	The location in the directory where groups/roles are stored.
<CIR_HOST>		`cir.example.com`	The fully qualified hostname of the Container Image Registry

ORACLE

Variable	Your Value	Sample Value	Description
<CIR_REPOSITORY>		`cir.example.com /repository/oaa`	The repository where the OAA images will be pushed to.
<USER_CERT_P12>		`/workdir/ cert.p12`	The location of the `<WORKDIR>/ cert.p12` on the *<INSTALL_HOST>*. This is only required if you generated third party certificates.
<USER_CERT_P12_PWD>		`password`	The password for the `cert.p12`. This is only required if you generated third party certificates.
<TRUST_CERT_P12>		`/workdir/ trust.p12`	The location of the `<WORKDIR>/ trust.p12` on the *<INSTALL_HOST>*. This is only required if you generated third party certificates.
<TRUST_CERT_P12_PWD>		`password`	The password for the `trust.p12`. This is only required if you generated third party certificates.

5.3.2 Editing the installOAA.properties

The following are the properties that must be changed in the `installOAA.properties` based on installing as per the Oracle recommended sandbox architecture in Supported Architectures. Properties not listed below should not be changed from their default values. Where variables are referenced, replace with your corresponding value.

> **Note:**
>
> If you are not using the Oracle recommended sandbox architecture Supported Architectures, you will need to refer to Understanding installOAA.properties Parameters, as the parameters to change may differ.

Common Deployment Configuration

Variable	Sample Value	Description
`common.deployment.keystorepassphrase=`*<USER_CERT_P12_PWD>*	`password`	If using your own certificates, you must set this to *<USER_CERT_P12_PWD>*. If you are going to use the self-signed certificates generated by OAA during installation, then set to a password of your choice.
`common.deployment.truststorepassphrase=`*<TRUST_CERT_P12_PWD>*	`password`	If using your own certificates, you must set this to *<TRUST_CERT_P12_PWD>*. If you are going to use the self-signed certificates generated by OAA during installation, then set to a password of your choice.

Variable	Sample Value	Description
`common.deployment.mode=<type>`	Both	Determines the installation type. Set `<type>` to the value based on the components you wish to install: • `Both` - install OAA and OARM. • `OUA` - install OAA, OARM, and OUA. • `OAA` - install OAA only.

Database Configuration

Variable	Sample Value	Description
`database.host=<DB_HOST>`	db.example.com	The fully qualified hostname of the database server where you want to store your OAA schemas.
`database.port=<DB_PORT>`	1521	The database listener port.
`database.syspassword=<SYS_PWD>`	password	The password of the SYS user in the database.
`database.schemapassword=<passwo rd>`	password	The password you want to set for the OAA schema in the database
`database.svc=<DB_SERVICE>`	orcl.example.co m	The database service name.
`database.name=<DB_NAME>`	orcl	The database name.

OAUTH Configuration

Variable	Sample Value	Description
`oauth.identityprovider=<IDS TORE>`	OUDStore	The Default User Identity Store used by OAM.
`oauth.redirecturl=<WEB_HOST >`	https:// ohs.example.com	The OHS URL used as the entry point to OAM. If a load balancer front ends the OHS then this value is the load balancer *<LBR_HOST>*.
`oauth.applicationid=default`	default	Application ID for OAA. Can be set to any value.
`oauth.adminurl=<WEB_HOST>`	https:// ohs.example.com	The OHS URL used as the entry point to OAM. If a load balancer front ends the OHS then this value is the load balancer *<LBR_HOST>*.
`oauth.basicauthzheader=<OAM _ADMIN_BASE64>`	b2FtYWRtaW46cGFzc3d vcmQ=	The BASE64 encoded value for *<OAM_ADMIN_USER>*:*<OAM_ADMIN_PAS SWORD>*
`oauth.identityuri=<WEB_HOST >`	https:// ohs.example.com:443	The OHS URL used as the entry point to OAM. If a load balancer front ends the OHS then this value is the load balancer URL *<LBR_HOST>*. For this value only, if the port is the default SSL port (443), you must append the port to the URL.

Variable	Sample Value	Description
`oauth.clientpassword=<passw ord>`	password	Set to a password of your choice. This will be the password for the OAuth client created during OAA installation.

Vault Configuration

Variable	Sample Value	Description
`vault.fks.server=<NFS_HOST>`	nfs.example.com	The fully qualified hostname of the NFS Server used by the Kubernetes cluster.
`vault.fks.path=<NFS_VAULT_P ATH>`	/nfs/mountOAApv/ OAAVault	The path on the NFS server to the *<NFS_VAULT_PATH>*
`vault.fks.key=<base64_passw ord>`	cGFzc3dvcmQ=	Set to a password of your choice. The password must be BASE64 encoded. To find the BASE64 encoded version run: `echo -n <password>	base64`

Chart Configuration

Variable	Sample Value	Description
`install.global.repo=<CIR_RE POSITORY>`	cir.example.com/ repository/oaa	The repository where the OAA images will be pushed to.
`install.global.image.tag=12 .2.1.4.1-<YYMMDD>`	12.2.1.4.1-<YYMMDD>	The image tag for the release.
`install.global.uasapikey=<p assword>`	password	Set to a password of your choice.
`install.global.policyapikey =<password>`	password	Set to a password of your choice.
`install.global.factorsapike y=<password>`	password	Set to a password of your choice.
`install.global.riskapikey=< password>`	password	Set to a password of your choice.
`install.global.drssapikey=< password>`	password	Set to a password of your choice. This only needs to be set if you have set `common.deployment.mode=OUA.`

Optional Configuration

Variable	Sample Value	Description
`install.global.ingress.enab led=true`	true	Must be set to true when using your own ingress controller.

ORACLE

Variable	Sample Value	Description
`install.global.serviceurl=<`_`WEBHOST>`_	`https://` `ohs.example.com`	The OHS URL used as the entry point to OAM. If a load balancer front ends the OHS then this value is the load balancer URL _<LBR_HOST>_. This parameter is used for the OAA runtime URL's.
`install.oaa-admin-` `ui.serviceurl=<`_`WEB_HOST>`_	`https://` `ohs.example.com`	The OHS URL used as the entry point to OAM. If a load balancer front ends the OHS then this value is the load balancer URL _<LBR_HOST>_. This parameter is used for the OAA Administration URL's.

OAA Management Configuration

Variable	Sample Value	Description
`install.mount.config.path=<`_`NFS_`_ _`CONFIG_PATH>`_	`/nfs/mountOAApv/` `OAAConfig`	The path on NFS server to the _<NFS_CONFIG_PATH>_.
`install.mount.config.server=<`_`NF`_ _`S_HOST>`_	`nfs.example.com`	The fully qualified hostname of the NFS Server used by the Kubernetes cluster.
`install.mount.creds.path=<`_`NFS_C`_ _`REDS_PATH>`_	`/nfs/mountOAApv/` `OAACreds`	The path on NFS server to the _<NFS_CREDS_PATH>_.
`install.mount.creds.server=<`_`NFS`_ _`_HOST>`_	`nfs.example.com`	The fully qualified hostname of the NFS Server used by the Kubernetes cluster.
`install.mount.logs.path=<`_`NFS_LO`_ _`GS_PATH>`_	`/nfs/mountOAApv/` `OAALogs`	The path on NFS server to the _<NFS_LOGS_PATH>_.
`install.mount.logs.server=<`_`NFS_`_ _`HOST>`_	`nfs.example.com`	The fully qualified hostname of the NFS Server used by the Kubernetes cluster.
`common.local.sslcert=<`_`WORKDIR>`_`/` `cert.p12`	`/workdir/cert.p12`	The location of the _<WORKDIR>/_`cert.p12` on the _<INSTALL_HOST>_. This parameter is only required if you generated third party certificates in Generating Server Certificates and Trusted Certificates.
`common.local.trustcert=<`_`WORKDIR`_ `>/trust.p12`	`/workdir/trust.p12`	The location of the _<WORKDIR>/_`trust.p12` on the _<INSTALL_HOST>_. This parameter is only required if you generated third party certificates in Generating Server Certificates and Trusted Certificates.

OUA Configuration

The parameters in this section only have to be set if you are deploying OUA.

Variable	Sample Value	Description
`oua.tapAgentFilePass=`*`<OUA_TAPFILE_PASSWORD>`*	`cGFzc3dvcmQ=`	The BASE64 encoded password of the `OAMOUAKeyStore.jks`.
`oua.tapAgentFileLocation=`*`<OUA_TAPFILE_LOCATION>`*	`/workdir/OAMOUAKeyStore.jks`	The location of the *`<WORKDIR>`*`/OAMOUAKeyStore.jks` on the *`<INSTALL_HOST>`*.
`oua.oamRuntimeEndpoint=`*`<WEB_HOST>`*	`https://ohs.example.com`	The OHS URL used as the entry point to OAM. If a load balancer front ends the OHS then this value is the load balancer URL *`<LBR_HOST>`*.

LDAP Configuration

Variable	Sample Value	Description
`ldap.server=`*`<LDAP_SERVER>`*	`ldap://oud.example.com:1389`	The LDAP server protocol, hostname and port.
`ldap.username=`*`<LDAP_ADMIN_USER>`*	`cn=oudadmin`	The user name of the directory administrator.
`ldap.password=`*`<LDAP_ADMIN_PWD>`*	`password`	The password of the directory administrator.
`ldap.oaaAdminUser=cn=oaaadmin,`*`<LDAP_USER_SEARCHBASE>`*	`cn=oaaadmin,cn=Users,dc=example,dc=com`	The OAA administration user to be created in the LDAP user search base.
`ldap.adminRole=cn=OAA-Admin-Role,`*`<LDAP_GROUP_SEARCHBASE>`*	`cn=OAA-Admin-Role,cn=Groups,dc=example,dc=com`	The OAA-Admin-Role group to be created in the LDAP group search base.
`ldap.userRole=cn=OAA-App-User,cn=`*`<LDAP_GROUP_SEARCHBASE>`*	`cn=OAA-App-User,cn=Groups,dc=example,dc=com`	The OAA-App-User group to be created in the LDAP group search base.
`ldap.oaaAdminUserPwd=<password>`	`password`	Set to a password of your choice. This will be the password for the `oaaadmin` user.
`ldap.addExistingUsers=<yes/no>`	`yes`	Set this value to `yes` if you want the OAA installation to add all your existing users in your *`<LDAP_USER_SEARCHBASE>`* to the OAA-App-User group. See Creating Users and Groups in the LDAP Store for more details.

OAA Configuration

Variable	Sample Value	Description
`oaa.tapAgentFilePass=`*`<OAA_TAPFILE_PASSWORD>`*	`cGFzc3dvcmQ=`	The BASE64 encoded password of the `OAMOAAKeyStore.jks`.

Variable	Sample Value	Description
`oaa.tapAgentFileLocation=<OAA_T APFILE_LOCATION>`	/workdir/ OAMOAAKeyStore. jks	The location of the `<WORKDIR>/` OAMOAAKeyStore.jks on the `<INSTALL_HOST>`.

Additional Considerations

If you have pushed the additional image (`oraclelinux:8-slim`) referenced in Setting Up a Container Image Registry (CIR) to a repository, you must add the following parameter to the **## 5. Chart configuration#** section:

> **✎ Note:**
>
> This parameter is not referenced in the default file so you must add it.

```
install.global.testrepo=<CIR_REPOSITORY>
```

Next Steps: Creating the Management Container

5.4 Creating the Management Container

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

1. On the *<INSTALL_HOST>*, navigate to the `<WORKDIR>/oaaimages/oaa-install` directory. For example:

   ```
   cd /workdir/oaaimages/oaa-install
   ```

2. Run the `installManagementContainer.sh` script. 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	Yes	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.
-E	No	Specifies if the Oracle Database is configured for SSL and you want to encrypt traffic between OAA and the database using SSL. Before using this parameter you must follow the **Configuring SSL between OAA and the Database** section in Database Configuration.
-s	No	Used in conjunction with the `-E` argument. Specifies the path to the `wallet.zip`. Before using this parameter you must follow the **Configuring SSL between OAA and the Database** section in Database Configuration.
-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.

ORACLE

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 the http_proxy, https_proxy variables set in the 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. For example:
	
	`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 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>
NAMESPACE: oaans
STATUS: deployed
REVISION: 1
TEST SUITE: None
Waiting 15 secs for OAA mgmt deployment to start...
Executing 'kubectl get pods oaamgmt-oaa-mgmt-84955fdf8f-x22k4 -n oaans'...
NAME                                  READY   STATUS             RESTARTS
AGE
oaamgmt-oaa-mgmt-84955fdf8f-x22k4   0/1     ContainerCreating   0
15s
Waiting 15 secs for OAA mgmt deployment to run...
Executing 'kubectl get pods oaamgmt-oaa-mgmt-84955fdf8f-x22k4 -n oaans'...
NAME                                  READY   STATUS             RESTARTS
AGE
oaamgmt-oaa-mgmt-84955fdf8f-x22k4   0/1     ContainerCreating   0
30s
Waiting 15 secs for OAA mgmt deployment to run...
Executing 'kubectl get pods oaamgmt-oaa-mgmt-84955fdf8f-x22k4 -n oaans'...
NAME                                  READY   STATUS             RESTARTS
AGE
```

```
oaamgmt-oaa-mgmt-84955fdf8f-x22k4   0/1      ContainerCreating   0
45s
Waiting 15 secs for OAA mgmt deployment to run...
Executing 'kubectl get pods oaamgmt-oaa-mgmt-84955fdf8f-x22k4 -n oaans'...
NAME                                   READY   STATUS              RESTARTS
AGE
oaamgmt-oaa-mgmt-84955fdf8f-x22k4   0/1      ContainerCreating   0
60s
Waiting 15 secs for OAA mgmt deployment to run...
Copying the OUA Agent jks file: /u01/oracle/scripts/creds/
OAMOUAKeyStore.jks
Copying the OAA Agent jks file to: /u01/oracle/scripts/creds/
OAMOAAKeyStore.jks
Copying OAA properties file to oaans/oaamgmt-oaa-mgmt-84955fdf8f-
x22k4:/u01/oracle/scripts/settings
Generating kube config for OAA mgmt pod 'oaans/oaamgmt-oaa-mgmt'.
Using service account 'oaans/oaamgmt-oaa-mgmt'.
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-84955fdf8f-
x22k4:/u01/oracle/scripts/creds...
Using helm config '/home/opc/.config/helm/repositories.yaml'.
Copying helm config to oaans/oaamgmt-oaa-mgmt-84955fdf8f-x22k4:/u01/oracle/
scripts/creds/helmconfig
Use command 'kubectl exec -n oaans -ti oaamgmt-oaa-mgmt-84955fdf8f-x22k4
-- /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-84955fdf8f-x22k4 -- /bin/bash
```

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

```
[oracle@oaamgmt-oaa-mgmt-84955fdf8f-x22k4 /]$
```

> **Note:**
>
> If you encounter problems creating the Management Container, see Troubleshooting the Installation.

Next Steps: Deploying OAA, OARM, and OUA

5.5 Deploying OAA, OARM, and OUA

Before deploying OAA, OARM, and OUA, make sure you have followed Creating the Management Container.

Deploying OAA, OARM, and OUA

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-84955fdf8f-x22k4 -- /bin/bash
```

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

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

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

During installation the deployment progress is shown on the screen. If you need more detailed information, you can view the `install.log`. The log is accessible from within the management container at `/u01/oracle/logs`, or outside the container at *<NFS_LOGS_PATH>*.

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

> **Note:**
>
> If you encounter problems deploying, see Troubleshooting the Installation.

5.6 Printing Deployment Details

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

> **Note:**
>
> Assuming the sandbox environment, the URL's will use the *<WEB_HOST>*. If using a load balancer in front of OHS, it will use the *<LBR_HOST>*.

```
############################OAA Deployment Details:
START####################################
OAAService=https://ohs.example.com/oaa/runtime
AdminUrl=https://ohs.example.com/oaa-admin/index.html
PolicyUrl=https://ohs.example.com/oaa-policy
SpuiUrl=https://ohs.example.com/oaa/rui
Email=https://ohs.example.com/oaa-email-factor
Push=https://ohs.example.com/oaa-push-factor
Fido=https://ohs.example.com/fido
SMS=https://ohs.example.com/oaa-sms-factor
TOTP=https://ohs.example.com/oaa-totp-factor
YOTP=https://ohs.example.com/oaa-yotp-factor
KBA=https://ohs.example.com/oaa-kba
RELEASENAME=oaainstall
# Key below is Base64 encoded API key
oaaapikey=cGFzc3dvcmQ=
# Key below is Base64 encoded Policy API key
oaapolicyapikey=cGFzc3dvcmQ=
# Key below is Base64 encoded Factor API key
oaafactorapikey=cGFzc3dvcmQ=
############################Deployment Details: END####################################
```

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

```
#############################Risk Deployment Details:
START####################################
AdminUrl=https://ohs.example.com/oaa-admin/index.html
PolicyUrl=https://ohs.example.com/oaa-policy
RISK=https://ohs.example.com/risk-analyzer
RISKCC=https://ohs.example.com/risk-cc
RELEASENAME=oaainstall
# Key below is Base64 encoded Policy API key
oaapolicyapikey=cGFzc3dvcmQ=
# Key below is Base64 encoded Factor API key
riskfactorapikey=cGFzc3dvcmQ=
#############################Deployment Details:
END####################################
```

For OUA installations, the DRSS Deployment Details are also printed on the console:

```
#####DRSS Deployment Details: START#####
DRSS=https://ohs.example.com/oaa-drss
RELEASENAME=oaainstall
# Key below is Base64 encoded DRSS API key
drssapikey=cGFzc3dvcmQ=
##################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
Administration Console	`AdminUrl`	`https://<hostname.domain>:<port>/oaa-admin/index.html`	The administrator username, `oaaadmin`.	`<password>` set in OAM OAuth identity store.
Self-Service Portal	`SpuiUrl`	`https://<hostname.domain>:<port>/oaa/rui`	Username from OAM OAuth identity store. The username must exist in the **OAA-App-User** group. See Creating Users and Groups in the LDAP 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/index.html` 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` `<RELEASENAME>` can be in lowercase.	`<Base64Decoded(oaapolicyapikey)>`
OAA Runtime	`OAAService`	`https://<hostname.domain>:<port>/oaa/runtime`	`<RELEASENAME>-oaa` `<RELEASENAME>` can be in lowercase.	`<Base64Decoded(oaaapikey)>`

ORACLE

REST API	Print Details Reference **	URL	Username **	Password **
Risk	`RISK`	`https://<hostname.domain>:<port>/risk-analyzer`	`<RELEASENAME>-risk` `<RELEASENAME>` can be in lowercase.	`<Base64Decoded(risk factorapikey)>`
Risk Customer Care	`RISKCC`	`https://<hostname.domain>:<port>/risk-cc`	`<RELEASENAME>-risk-cc` `<RELEASENAME>` can be in lowercase.	`<Base64Decoded(risk factorapikey)>`
KBA	`KBA`	`https://<hostname.domain>:<port>/oaa-kba`	`<RELEASENAME>_OAA_KBA` The `<RELEASENAME>` must be all uppercase. If `<RELEASENAME>` contains "-", then replace with "_". For example `oaa-install` translates to : `OAA_INSTALL_OAA_KBA`.	`<Base64Decoded{oaafactorsapikey}>`
DRSS	`DRSS`	`https://<hostname.domain>:<port>/oaa-drss`	`<RELEASENAME>_OAA_DRSS` The `<RELEASENAME>` must be all uppercase. If `<RELEASENAME>` contains "-", then replace with "_". For example oaa-install translates to : `OAA_INSTALL_OAA_DRSS`.	`<Base64Decoded(drss apikey)>`

** 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`**, for** example `oaainstall-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

Next Steps: Post Installation Steps for Oracle Universal Authenticator

5.7 Post Installation Steps for Oracle Universal Authenticator

Follow these post installation steps if you installed OAA with Oracle Universal Authenticator (OUA).

1. Login to the Oracle Access Management Administration console.

2. In the **Application Security** Launchpad, click **Application Domains**.

3. In the **Search Application Domains** screen, click **Search**.

4. In the **Search Results**, click **IAM Suite**.

5. In the **IAM Suite** screen, click the **Authorization Policies** tab.

6. In the **Authorization Policies** tab, click **Protected Resource Policy**.

7. In the **Protected Resource Policy** screen, click the **Responses** tab.

8. Click **Add** to add a new response.

9. In the **Add Response** screen select the following values and click **Add**:

    - **Type**: `Session`

    - **Name**: `allowPersistentLogin`

    - **Value**: `true`

10. Click **Apply**.

5.8 Troubleshooting the Installation

This section provides troubleshooting tips for installing OAA, OARM, and OUA.

- Problems Running installManagementContainer.sh

- Problems Running OAA.sh

5.8.1 Problems Running installManagementContainer.sh

This section provides troubleshooting tips for problems while running `installManagementContainer.sh`.

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.

5.8.2 Problems Running OAA.sh

This section provides troubleshooting tips for problems while running `OAA.sh`.

General failures during OAA.sh

If the `OAA.sh` deployment fails at any stage, you can generally fix the underlying issue and then rerun `OAA.sh`. Any configuration tasks already performed by the installation will be skipped when OAA.sh is rerun.

Log information

If the deployment fails while running `OAA.sh` and you need more detailed information, you can view the `install.log`. The log is accessible from within the management container at `/u01/oracle/logs`, or outside the container at *<NFS_LOGS_PATH>*.

If the install.log references another component log file, this log can also be found in the same location.

Trust and Cert Store Configuration failed

If you receive the following message:

```
Configuring Trust and Cert Store for OAA.
Trust and Cert Store Configuration failed. Check log /u01/oracle/logs/
install.log for details.
```

and the `install.log` shows:

```
Configuring Trust and Cert Store for OAA.
Checking oauth.identityuri mentioned in /u01/oracle/scripts/settings/
installOAA.properties
 Property oauth.identityuri is https, will proceed to download certificate
for https://ohs.example.com
 Checking url connectivity for https://ohs.example.com -
139784391608128:error:02002071:system library:connect:No route to
host:crypto/bio/b
_sock2.c:110: 139784391608128:error:2008A067:BIO routines:BIO_connect:connect
error:crypto/bio/b_sock2.c:111: connect:errno=113
Failed
```

This means that either:

- The URL being accessed, for example `https://ohs.oracle.com`, is not accessible; or

- The OHS (or load balancer) being accessed is using the default 443 port, and the `oauth.identityuri` parameter in `installOAA.properties` does not have `:443` appended. See, Editing the installOAA.properties.

Resolve the problem by editing the *<NFS_CONFIG>*/`installOAA.properties` and run `OAA.sh` again.

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 Configuring OAuth and Oracle HTTP Server.

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 Configuring OAuth and Oracle HTTP Server.

Configuring OAM for OAA. OAM for OAA setup failed

If you receive the following message:

```
Configuring OAM for OAA. OAM for OAA setup failed. Check log /u01/oracle/logs/
install.log for details.
```

The `install.log` may refer you to the `/u01/oracle/logs/add_resources.log`. In the `add_resources.log` you may see:

```
curl -sk --connect-timeout 30  -X POST 'https://ohs.example.com/oam/services/
rest/11.1.2.0.0/ssa/policyadmin/resource'  -H 'Content-Type: application/
json' -H 'Authorization: Basic <Base64EncodedUser:Password>' -d
'{"queryString":null,"applicationDomainName":"IAM
Suite","hostIdentifierName":"IAMSuiteAgent","resourceURL":"/oauth2/rest/
**","protectionLevel":"EXCLUDED","QueryParameters":null,"resourceTypeName":"HT
TP","Operations":null,"description":"/oauth2/rest/**","name":"/oauth2/rest/
**","id":"1"}'
```

If no error is referenced, then run the command from inside the management container. You will need to substitute `<Base64EncodedUser:Password>` with the value set for the parameter `oauth.basicauthzheader` in the `installOAA.properties`. If you receive the error `Error 401--Unauthorized`, then this means the value set for `oauth.basicauthzheader` is incorrect. Edit the `<NFS_CONFIG>/installOAA.properties` and set the parameter to the correct value. Then rerun `OAA.sh`. See, Editing the installOAA.properties for more details.

Problems starting pods

During OAA.sh, when it gets to the `Installing OAA` section displayed in the output, you can check the status of the pods by running:

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

If any of the pods fail or do not start, you can run the following commands to get more details:

```
kubectl logs -f <pod> -n oaans
kubectl describe pod <pod> -n oaans
```

During OAA.sh pods fail to start and show CrashLoopBackOff

Run the `kubectl logs <pod> -n <namespace>` 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 `https://ohs.example.com/.well-known/openid-configuration` because the `PathTrim` and `PathPrepend` in the `mod_wl_ohs.conf` for that entry were not updated. See Configuring OAuth and Oracle HTTP Server.

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.

Failed to import snapshot

If you receive the following message during installation:

```
Importing the snapshot file : /u01/oracle/scripts/oarm-12.2.1.4.1-base-
snapshot.zip
Executing CURL : curl --silent -k --location --request POST 'https://
ohs.example.com/policy/risk/v1/snapshots'
      --header 'Authorization: Basic
b2FhaW5zdGFsbC1vYWEtcG9saWN5OldlbGNvbWUx'      --header 'Content-type: ap
plication/octet-stream'      --data-binary '@/u01/oracle/scripts/
oarm-12.2.1.4.1-base-snapshot.zip'
Import status : {"status":"201","message":"Snapshot created successfully.
","snapshot":{"name":"OARM Snapshot","description":"OARM Snapshot",
"snapshotId":"1","createTime":"10-10-2024 16:32:45"}}
Upload status : 201
Snapshot ID : 1
Applying snapshot : curl --silent -k --location --request POST 'https://
ohs.example.com/policy/risk/v1/snapshots/1/apply'      --header
'Authorization: Basic b2FhaW5zdGFsbC1vYWEtcG9saWN5OldlbGNvbWUx'
Apply result : <html>
<head><title>504 Gateway Time-out</title></head>
<body>
<center><h1>504 Gateway Time-out</h1></center>
<hr><center>nginx</center>
</body>
</html>
parse error: Invalid numeric literal at line 1, column 7
Fail to apply the snapshot: 1
Failed to import snapshot.
```

Then it is possible the ingress controller is timing out before it has had a chance to complete the snapshot import. One possible solution to this is update the ingress controller as follows:

```
kubectl annotate ingress -n oaans nginx.ingress.kubernetes.io/proxy-read-
timeout=3600
```

```
nginx.ingress.kubernetes.io/proxy-connect-timeout=3600
nginx.ingress.kubernetes.io/proxy-send-timeout=3600 --all
```

Error 'jq: error: Invalid escape at line 1, column 6` Creating tap partner in OAA

If you see the following error running `OAA.sh` then the `oua.tapAgentFilePass` value was not was not set in base64:

```
jq: error: Invalid escape at line 1, column 6 (while parsing '"\ï¿½"') at
<top-level>, line 1:
.agentName |= if . == "" then "MFAOAAPartner17ohsapr9" else . end
|            .privateKey |= if . == "" then
"CECECECE0000000200000001..etc..
jq: 1 compile error
Creating tap partner in OAA
```

To solve this problem, set the value to the base64 version of the password and run the `OAA.sh` again. See, Preparing the Properties file for Installation.

Bad Oracle Access Manager Request in DRSS Logs

If you see the following error in the DRSS pod logs:

```
<DATE> Thread[http-thread-34,5,server]: INFO
oracle.security.am.drss.handler.oam.OAMHandler parseOAMResponse Exception
during parseOAMResponse Unexpected character ('<' (code 60)): expected a
valid value (JSON String, Number, Array, Object or token 'null', 'true' or
'false')
 at [Source: (String)"<html><head><title>Bad Oracle Access Manager Request</
title><META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8"></
head><body><h1>Bad Oracle Access Manager Request</h1><p>Unable to process the
request due to unexpected error.</p></body></html>
```

Then the `oua.oamRuntimeEndpoint` was either set incorrectly in the `installOAA.properties`, not set to the fully qualified hostname of the OAM server, or the OAM server is not functioning correctly.

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 oamadmin:<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>'
```

5.9 Cleaning Up Installation

Perform the following steps to clean up an OAA, OARM, and OUA 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. Outside the container, run:

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

If any pods remain then run:

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

4. 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 oamadmin:<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>'
   ```

5. Login to the Oracle Access Management Administration console and perform the following tasks:

   a. In the **Application Security** Launchpad, click **Application Domains**.

   b. In the **Search Application Domains** screen, click **Search**.

   c. In the **Search Results**, click **IAM Suite**.

   d. Click **Resources** and **Search**

   e. For any resources that use the **OAA_MFA-Policy**, edit the resource and choose a different policy. Click **Apply**.

   f. Click on the **Authentication Policies** tab and delete the **OAA_MFA-Policy**. Click **Delete** to confirm.

ORACLE

g. In the **Application Security** Launchpad, click **Authentication Schemes**. Click **Search**. Delete the **OAA-MFA-Scheme** and click **Delete** to confirm.

h. In the **Application Security** Launchpad, click **Authentication Modules**. Click **Search**. Delete the **OAA-MFA-Authn-Module** and click **Delete** to confirm.

6. Drop the database schemas as follows, Replace `<OAA_RCU_PREFIX>` with your OAA schema. You can find the schema name used for the installation in the `installOAA.properties`, for example `database.schema=DEV_OAA`:

```
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.

Part III
Upgrading OAA, OARM, and OUA

6

Upgrading OAA, OARM, and OUA

Upgrade OAA, OARM, and OUA to the latest release.

To upgrade to the latest release follow the sections below:

- Preparing the Environment for Upgrade
- Performing the Upgrade

6.1 Preparing the Environment for Upgrade

Before starting an upgrade, you must prepare your existing environment.

General Prerequisites

The following must be met before performing the upgrade:

- 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.
- Make sure the installation host prerequisites are met as per Installation Host Requirements.
- The installation host must have access to the NFS volumes for the existing deployment. See Configuring NFS Volumes.
- Backup file system data, runtime data, and policy and configuration data. See, Backing Up OAA/OARM.

Upgrading To Include Oracle Universal Authenticator

If you haven't previously configured Oracle Universal Authenticator (OUA) and want to upgrade OAA to include OUA, you must perform the following steps:

1. Upgrade Oracle Access Management (OAM) to April 24 Bundle Patch or later.
2. Follow Oracle Access Management Requirements and perform any prerequisite steps that are required for OUA, for example missing OHS entries, and OUA TAP partner.
3. Add an additional image registry, `oaa-drss`, to your repository. See Setting Up a Container Image Registry (CIR).
4. Add the LDAP attribute *obpsftid: true* to any user who is to use OUA. See Creating Users and Groups in the LDAP Store.

Push Notification for Android Devices

The following section is only relevant if you are upgrading from any release prior to June 24 and are already using Push Notifications. If you are already using the June 24 release or not using Push Notifications, you can ignore this section.

From June 2024 Google are removing deprecated FCM legacy APIs, see Migrate from legacy FCM APIs to HTTP v1. As a result, if you have push notifications for Android configured in your existing deployment, you need to perform some changes to your existing configuration:

1. Login to Google Firebase console at https://console.firebase.google.com/.

2. Click on your existing Firebase project, for example `OAAAndroidPUSH`.

> **Note:**
>
> If you no longer have access to the account that created the project, you will need to create a new project using the instructions in Creating a Google Firebase Project Enabled for Google Cloud Messaging and then update the `senderId` parameter as per Configuring OAA Properties for Android Push Notification.

3. In the left navigation pane of the project window, click the **Settings** icon, and then select **Project settings**.

4. Go to the **Service Accounts** tab and click **Generate new private key**. In the **Generate new private key** window, click **Generate key**. This will generate a service account json file. Download and save the file, for example `service-account.json`, and keep it secure.

5. Create a directory in the NFS volume `<NFS_VAULT_PATH>` and copy the json file across:

```
cd <NFS_VAULT_PATH>
$ mkdir -p ChallengeOMAPUSH/gcm
$ cp service-account.json <NFS_VAULT_PATH>/ChallengeOMAPUSH/gcm
$ sudo chmod 444 <NFS_VAULT_PATH>/ChallengeOMAPUSH/gcm/service-account.json
```

6. Edit the `<NFS_LOG_PATH>/status.info` and set the following vault parameters to false, then save the file:

```
VAULTINSTALL=false
VAULTCHECK=false
```

Preparing the Environment for Upgrade

1. Take a backup of the existing `installOAA.properties` in case you need to rollback the upgrade:

```
cd <NFS_CONFIG_PATH>
cp installOAA.properties installOAA.properties.<release-date>
```

2. In the `<NFS_LOG_PATH>/status.info` make sure `SNAPSHOTIMPORT=true`.

> **Note:**
>
> This is a very important step. If this value is `false` make sure you change it to `true`, otherwise your OAA snapshot will be overwritten during the upgrade!

3. Download and extract the latest OAA installation files to the installation host as per Obtaining the Installation Software.

> **✎ Note:**
>
> When you unzip the file it will create the `oaa-install` directory, for
> example `$UPGRADE_WORKDIR/oaa-install`. This directory will be used in the steps
> that follow.

4. 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
```

5. Copy the `installOAA.properties` for the current deployment to the `$UPGRADE_WORKDIR/oaa-install` directory:

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

For example:

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

6. Edit the `$UPGRADE_WORKDIR/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 `$UPGRADE_WORKDIR/oaa-install/installOAA.properties.template`.

7. If you haven't previously installed OUA and want to upgrade your installation to include
   OUA, you must edit the `$UPGRADE_WORKDIR/oaa-install/installOAA.properties` and
   change the following:

   • `common.deployment.mode=OUA`

   • `install.global.drssapikey=drssapikeytobesetduringinstallation`

   > **✎ Note:**
   >
   > Change `drssapikeytobesetduringinstallation` to a value of your choice.

- Set the following parameters:
    - `oua.tapAgentName=OAM-OUA-TAP`
    - `oua.tapAgentFilePass=<OUA_TAPFILE_PASSWORD>`
    - `oua.tapAgentFileLocation=<OUA_TAPFILE_LOCATION>`
    - `oua.oamRuntimeEndpoint=<WEB_HOST>`

> **Note:**
>
> See Configuration Checkpoint for more details.

8. Add the following new section:

```
################################### OAA Configuration
###########################################
oaa.tapAgentName=<OAA_TAP_PARTNER_NAME>
oaa.tapAgentFilePass=<OAA_TAP_FILE_PASS>
oaa.tapAgentFileLocation=<LOCAL_PATH>/<LOCAL_TAP_AGENT_FILE>
oaa.authFactors=ChallengeEmail,ChallengeSMS,ChallengeOMATOTP,ChallengeYubic
oOTP,ChallengeOMAPUSH,ChallengeFIDO2
##################################################################################
########################
```

Administrators should be aware of the following:

- If you have previously integrated OAM with OAA as per Integrate Oracle Access Management with Oracle Advanced Authentication, then:
    - `oaa.tapAgentName` must be set to the name of the OAA TAP Partner for your existing deployment. The name can be found by referencing the tutorial above.
    - If the OAA TAP Partner is found during the upgrade then the `oaa.tapAgentFilePass`, `oaa.tapAgentFileLocation`, and `oaa.authFactors` are ignored. However, they must be set to either the default values listed above, or you can copy your TAP Partner keystore to a local directory on the *<INSTALL_HOST>* and set the parameters correctly.

- If you have not previously integrated OAM with OAA you should follow the **Registering OAA as a TAP Partner in OAM** section in Registering OAM TAP Partners, and set the parameters accordingly.

9. If you are upgrading from a release prior to June 24 and have Push notifications for Android configured, you must edit the `$UPGRADE_WORKDIR/oaa-install/installOAA.properties` file and update the `common.deployment.push.gcmjsonfile` parameter to point at the `service-account.json` file downloaded earlier:

```
common.deployment.push.gcmjsonfile=/u01/oracle/service/store/oaa/
ChallengeOMAPUSH/gcm/service-account.json
```

> ✏️ **Note:**
>
> /u01/oracle/service/store/oaa/ChallengeOMAPUSH/gcm/service-account.json **is the internal mapping to** <NFS_VAULT_PATH>/ChallengeOMAPUSH/gcm/service-account.json.

10. Add the following new section to the installOAA.properties. For details on these parameters see Creating Users and Groups in the LDAP Store:

```
################################### LDAP configuration
#########################################
ldap.server=<LDAP_SERVER>
ldap.username=<LDAP_ADMIN_USER>
ldap.password=<LDAP_ADMIN_PWD>
ldap.oaaAdminUser=cn=oaaadmin,<LDAP_USER_SEARCHBASE>
ldap.adminRole=cn=OAA-Admin-Role,<LDAP_GROUP_SEARCHBASE>
ldap.userRole=cn=OAA-App-User,cn=<LDAP_GROUP_SEARCHBASE>
ldap.oaaAdminUserPwd=<password>
ldap.addExistingUsers=no
```

11. 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
```

6.2 Performing the Upgrade

Upgrade the existing OAA installation to the latest version.

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

```
cd $UPGRADE_WORKDIR/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:

ORACLE

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".

`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. When asked `Would you like to perform an upgrade? [Y/N]`, enter `Y`.

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

```
...
Copying OAA properties file to oaans/oaamgmt-oaa-mgmt-7dbfbd87dc-
r7v29:/u01/oracle/scripts/settings
Use command 'kubectl exec -n oaans -ti oaamgmt-oaa-mgmt-7dbfbd87dc-r7v29
-- /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.
```

If there are any problems during upgrade, refer to Troubleshooting the Installation.

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

```
kubectl exec -n oaans -ti oaamgmt-oaa-mgmt-7dbfbd87dc-r7v29 -- /bin/bash
```

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

```
[oracle@oaamgmt-oaa-mgmt-7dbfbd87dc-r7v29 /]$
```

6. Inside the OAA management pod perform the upgrade:

```
[oracle@oaamgmt-oaa-mgmt-7dbfbd87dc-r7v29 /]$ cd ~
[oracle@7dbfbd87dc-r7v29 ~]$ ./OAA.sh -f installOAA.properties
```

During installation the deployment progress is shown on the screen. If you need more detailed information, you can view the `install.log`. The log is accessible from within the management container at `/u01/oracle/logs`, or outside the container at *<NFS_LOGS_PATH>*.

7. Once the upgrade is complete you should see the upgrade is successful and the deployment details are printed to the screen. See, Printing Deployment Details.
If there are any problems during the upgrade, refer to Troubleshooting the Installation.

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-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-drss-9f68f4856-849x4               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-7dbfbd87dc-r7v29                        1/1      Running
0          16m
```

9. If you have Push Notifications for Android configured and are upgrading from a release prior to June 24, check that push notifications work successfully. See Accessing a Protected Application Using Android Push Notification.

10. If you previously have used Archiving and Purging, then you must run `/u01/oracle/db_purge/archive/create_oaam_rebuild_index.sql` inside the management container.

   a. 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
   ```

   b. Inside the management container, navigate to the `/u01/oracle/db_purge/archive` directory and login to the OAA database as the OAA schema user, for example `DEV_OAA`:

   ```
   sqlplus <schema_name>/<password>@//db.example.com:1521/orcl.example.com
   ```

   c. Run the `create_oaam_rebuild_index.sql` script to update the rebuild index procedure:

   ```
   SQL> @create_oaam_rebuild_index.sql
   ```

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.

To rollback the OAA installation perform the following steps:

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

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

   For example:

   ```
   helm history oaamgmt -n oaans
   ```

   The output will look similar to the following:

   ```
   REVISION         UPDATED        STATUS
   CHART                                          APP VERSION     DESCRIPTION
   1                <date>         superseded     oaa-mgmt-1.0.0-12.2.1.4.1-
   <release-date>   0.1.0              Install complete
   2                <date>         superseded     oaa-mgmt-1.0.0-12.2.1.4.1-
   <relase-date+1>  0.1.0              Upgrade complete
   ```

2. Rollback the management container using the following command:

   > ✏️ **Note:**
   >
   > This command must be run outside the management container.

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

   For example:

   ```
   helm rollback oaamgmt -n oaans
   ```

3. Check the history of the OAA management charts to show the rollback has occurred:

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

   For example:

   ```
   helm history oaamgmt -n oaans
   ```

The output will look similar to the following:

```
REVISION        UPDATED       STATUS
CHART                                      APP VERSION     DESCRIPTION
1              <date>        superseded     oaa-mgmt-1.0.0-12.2.1.4.1-
<release-date>  0.1.0             Install complete
2              <date>        superseded     oaa-mgmt-1.0.0-12.2.1.4.1-
<release-date+1> 0.1.0            Upgrade complete
3              <date>        deployed       oaa-mgmt1.0.0-12.2.1.4.1-
<release-date>  0.1.0             Rollback to 1
```

4. Revert the `installOAA.properties` file to the version used before the upgrade took place:

```
cd <NFS_CONFIG_PATH>
cp installOAA.properties installOAA.properties.<release-date+1>
cp installOAA.properties.<release-date> installOAA.properties
```

5. Connect to the 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 /]$
```

6. Edit the `/u01/oracle/logs/status.info` file and update any properties that will affect the downgrade. For example, if the vault needs to be updated or recreated, set `VAULTINSTALL=false`. The properties you need to update will depend on what changes were made during that upgrade. If you are unsure contact Oracle Support for advice.

7. Run the following command inside the management container to update the OAA install:

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

8. Run the following command to confirm the installation was successfully downgraded:

```
 helm history <deployment-name> -n <namespace>
```

Where `<deployment-name>` is the value of the `common.deployment.name` in your `installOAA.properties`. For example:

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

The output will look similar to the following:

```
REVISION   UPDATED STATUS    CHART                                       APP
VERSION DESCRIPTION
2        <date>  superseded oaa-1.0.0-12.2.1.4.1-<release-date>
0.1.0       Upgrade complete
3        <date>  superseded oaa-1.0.0-12.2.1.4.1-<release-date+1>
0.1.0        Upgrade complete
```

```
4          <date>  deployed   oaa-1.0.0-12.2.1.4.1-<release-date>
0.1.0        Upgrade complete
```

9. Run the following command to check the pods are now using the previous image:

```
kubectl get deploy -o wide -n <namespace> |awk '{print $7}'
```

For example:

```
kubectl get deploy -o wide -n oaans |awk '{print $7}'
```

The output will look similar to the following:

```
IMAGES
      container-registry.example.com/repository/oaa-factor-
email:12.2.1.4.1-<release-date>
      container-registry.example.com/repository/oaa-factor-fido:12.2.1.4.1-
<release-date>
      container-registry.example.com/repository/oaa-svc:12.2.1.4.1-
<release-date>
      container-registry.example.com/repository/oaa-admin:12.2.1.4.1-
<release-date>
      container-registry.example.com/repository/oaa-drss:12.2.1.4.1-
<release-date>
      container-registry.example.com/repository/oaa-factor-kba:12.2.1.4.1-
<release-date>
      container-registry.example.com/repository/oaa-policy:12.2.1.4.1-
<release-date>
      container-registry.example.com/repository/oaa-factor-push:12.2.1.4.1-
<release-date>
      container-registry.example.com/repository/risk-engine:12.2.1.4.1-
<release-date>
      container-registry.example.com/repository/risk-cc:12.2.1.4.1-
<release-date>
      container-registry.example.com/repository/oaa-factor-sms:12.2.1.4.1-
<release-date>
      container-registry.example.com/repository/oaa-spui:12.2.1.4.1-
<release-date>
      container-registry.example.com/repository/oaa-factor-totp:12.2.1.4.1-
<release-date>
      container-registry.example.com/repository/oaa-factor-yotp:12.2.1.4.1-
<release-date>
      container-registry.example.com/repository/oaa-mgmt:12.2.1.4.1-
<release-date>
```

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

Features from OAAM that are not Supported in OARM and OAA

The following table lists the 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 Self-Service Portal.
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 the OAAM Environment for Transition
2. Prerequisite Configurations for Installing OAA, OARM, and OUA

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

3. Obtaining the Installation Software
4. Performing the Transition

9.1 Preparing the OAAM Environment for Transition

To transition an Oracle Adaptive Access Management (OAAM) environment to OAA, OARM, and OUA, 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)

> **Note:**
>
> If you intend to install Oracle Universal Authenticator (OUA) when you transition, you must be on the 12.2.1.4 April 24 Bundle Patch or later.

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, OARM, and OUA 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 The OAA, OARM, and OUA 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 Transition

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

Perform the following steps to transition from OAAM to OAA, OARM, and OUA.

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 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`, `Both`, or `OUA`. Default mode is `Both`, which installs OAA integrated with OARM.
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.

5. If you intend to install OUA, you must change the following in the `installOAA.properties`:

• `common.deployment.mode=OUA`

• `install.global.drssapikey=drssapikeytobesetduringinstallation.`

> **✎ Note:**
>
> Change `drssapikeytobesetduringinstallation` to a value of your choice.

• Edit the **OUA Configuration** section as per Oracle Universal Authenticator Configuration.

6. Deploy OAA, OARM, and OUA. For details, see Deploying OAA, OARM, and OUA.

7. 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

8. If you installed OUA, follow Post Installation Steps for Oracle Universal Authenticator.

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 the OAA, OARM, and OUA Environment

After transitioning to the OAA, OARM, and OUA 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

Administering Oracle Advanced Authentication

ORACLE®

11

Configuring Oracle Advanced Authentication

Topics

11.1 Onboarding Users in OAA

For end users to be able to access and use the Self-Service Portal, the user must be created in OAA.

Administrators have the following options to create users in OAA:

- Auto-create users using the Self-Service Portal.
- Use REST API's to create users and their factors.
- Use the OAAAuthnPlugin to migrate users from OAM.

The sections below outline the steps for each option.

Auto-create Users Using the Self-Service Portal

When an end user logs into the Self-Service Portal for the first time, the user will be created automatically in OAA. Once logged in to the Self-Service Portal, the end user can create their authentication factors manually.

If you have installed using the December 24 release or later, auto-creation of users is enabled out of the box.

If you have installed a release prior to OAA December 24 you need to configure this manually using the `oaa.default.spui.pref.runtime.autoCreateUser=true` property:

1. Set the property `oaa.default.spui.pref.runtime.autoCreateUser=true` using the `<PolicyUrl>/policy/config/property/v1` REST API endpoint.

> **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.

Use REST API's To Create Users and Their Factors

Administrators can create users and their factors using REST API's. Once the user is created via REST API's, they can log in to the Self-Service Portal and see all their authentication factors. Users can then manage their factors as they choose.

For more details, see Registering Users with Challenge Factors in OAA.

Use the OAAAuthnPlugin To Migrate Users From OAM

Administrators can use the OAAAuthnPlugin to migrate users and configured factors from Oracle Access Management (OAM). Once the OAAAuthnPlugin is configured, when a user accesses an OAM protected application, that user will atomically be migrated to OAA along with any factors configured, based on defined LDAP attributes.

> **Note:**
>
> Only Email, SMS, and Oracle Mobile Authenticator TOTP are supported for migration.

Once the user is migrated, the user can access the Self-Service Portal and view and manage their configured factors.

If you have installed using the December 24 release or later, then this integration is automatically configured for you.

If you have installed prior to December 24, then this is configured manually.

For details on how to configure the OAAAuthnPlugin manually, or for more information on how migrating users works, see Integrate Oracle Access Management with Oracle Advanced Authentication .

11.2 Creating Integration Agents in OAA

You must create integration agents to integrate client applications with OAA.

The following integration agents can be created in OAA:

- OAM Integration Agents
- REST API Integration Agents

- Oracle RADIUS Integration Agents

You can create integration agents using either REST APIs, or the OAA Administration Console.

For details about creating integration agents using REST APIs, see REST API for Administration in Oracle Advanced Authentication.

OAM Integration Agents

An OAM integration agent is created and configured for you during installation (assuming December 24 release or later).

For releases prior to December 24, you can 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**. For more information, see Register OAA as a TAP Partner in OAM .

    g. **Private key Password**: Specify the password that you had provided while registering OAM as a TAP partner of OAA.

4. Click **Save**

REST API Integration Agents

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**.

Oracle RADIUS Integration Agents

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

11.3 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**.

11.4 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.

11.5 Creating Groups in OAA

Type Value Reference for Groups

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**.

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.
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.

11.6 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 Self-Service Portal. For more information, see Managing Factors in the Self-Service Portal.

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 Self-Service Portal. See, Managing Factors in the Self-Service Portal.

> **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.

11.7 Managing Factors in the Self-Service Portal

OAA provides users with a Self-Service Portal for managing factors.

Access the Self-Service Portal 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 **My Authenticators** 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**, or **Delete** a factor, or set your default authentication factor. To do this, click the ellipsis button on the factor tile and select one of the following options:

- **Disable**: The factor is disabled, and will not be displayed during the second-factor authentication.

- **Set As Default**: The 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 Self-Service Portal.

- **Delete**: The factor is deleted.

- **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, the following factors can be 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.
	✎ **Note:** It is not possible to configure SafeID using Oracle Mobile Authenticator in the Self-Service Portal. SafeID configuration must be performed by an Administrator. See Configuring SafeID Challenge in Oracle Advanced Authentication.
Email Challenge	**Friendly Name**: Specify a name.
	Email: Specify the required email.
FIDO2 Challenge	**Friendly Name**: Specify a name.
	Click Register and then perform the required action for your FIDO2 device. For example if using Yubikey touch your Yubikey, or if using Windows Hello using fingerprint for example, touch your fingerprint.
	✎ **Note:** Depending on your FIDO2 implementation you may be asked to setup a security passkey before performing the specific action for your FIDO2 device.
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.

Factors	Values
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 these factors, see the following tutorials:

- Configuring SMS Challenge in the Oracle Advanced Authentication Self-Service Portal
- Configuring Email Challenge in the Oracle Advanced Authentication Self-Service Portal
- Configuring Mobile Authenticator Challenge in the Oracle Advanced Authentication Self-Service Portal
- Configuring Security Questions Challenge in the Oracle Advanced Authentication Self-Service Portal.
- Configuring FIDO2 Challenge in the Oracle Advanced Authentication Self-Service Portal.
- Configuring YubiKey Challenge in the Oracle Advanced Authentication Self-Service Portal.
- Configuring Push Notification Challenge with Oracle Mobile Authenticator in the Oracle Advanced Authentication Self-Service Portal
- Configuring SafeID Challenge in Oracle Advanced Authentication

11.8 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 Passcode (OTP), for example `oaa@example.com`

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

11.9 Configuration Properties for OAA

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

Configuration Properties for OAA

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.

Property Name	Default Value	Description
`bharosa.uio.default.all.factor.challengecounter.expiryTime`	1800000	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.
`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).

Property Name	Default Value	Description
`bharosa.uio.default.challe nge.type.enum.ChallengeEma il.msgResourceURLTemplate`	`Resource URL Access:`	Part of the email template to display message resource URL.
`bharosa.uio.default.challe nge.type.enum.ChallengeEma il.msgSubject`	`One Time Pin: OAA`	Subject title of the email template.
`bharosa.uio.default.challe nge.type.enum.ChallengeEma il.challengeCounterExpiryT ime`	`1800000`	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.fa ctor.challengecounter.expi ryTime` is used (default is 1800000 milliseconds) .
`bharosa.uio.default.challe nge.type.enum.ChallengeEma il.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.fa ctor.retry.count` is used (default is 10).
`bharosa.uio.default.challe nge.type.enum.ChallengeEma il.msgTimeTemplate`	`Time of Access:`	Part of the email template to display message time.
`bharosa.uio.default.challe nge.type.enum.ChallengeEma il.promptmessage`	`Send OTP to {0}`	Prompt message to send One Time Pin (OTP) through email used on end-user challenge page.
`bharosa.uio.default.challe nge.type.enum.ChallengeEma il.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.challe nge.type.enum.ChallengeOMA TOTP.challengeText`	`Enter OTP from device {1}`	Prompt message to enter time-based One Time Pin (OTP) on end-user challenge page.
`bharosa.uio.default.challe nge.type.enum.ChallengeOMA TOTP.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.

ORACLE®

Property Name	Default Value	Description
`bharosa.uio.default.challe nge.type.enum.ChallengeOMA TOTP.challengeCounterExpir yTime`	1800000	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.fa ctor.challengecounter.expi ryTime` is used (default is 1800000 milliseconds).
`bharosa.uio.default.challe nge.type.enum.ChallengeOMA TOTP.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.fa ctor.retry.count` is used (default is 10).
`bharosa.uio.default.challe nge.type.enum.ChallengeOMA TOTP.registration.showSecr etKeyText`	true	Displays a secret key in the Self-Service Portal, 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.challe nge.type.enum.ChallengeOMA TOTP.registration.showQrco de`	true	Displays a QR code in the Self-Service Portal, for use with Oracle Mobile Authenticator, Google Authenticator, or Microsoft Authenticator. If the value is set to `false`, the QR code isn't displayed.
`bharosa.uio.default.challe nge.type.enum.ChallengeOMA TOTP.keyExpiryEnabled`	false	A boolean value that indicates whether or not secret key expiration is enabled. When enabled, the Time-based One Time Passcode (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.challe nge.type.enum.ChallengeOMA TOTP.keyExpiryTimeMinutes`	60	Specifies the key's expiration time in minutes. This must be a positive whole number.
`bharosa.uio.default.challe nge.type.enum.ChallengeOMA TOTP.registration.otpexpir ytimeMs`	300000	Specifies the timeout in millisecond for the Time-based One Time Passcode (TOTP) generated registration URL.

ORACLE

Property Name	Default Value	Description
`bharosa.uio.default.challe nge.type.enum.ChallengeOMA TOTP.registration.oma.conf ig`	`oraclemobileauthenticator: //settings? ServiceName::=%deviceName% &ServiceType::=SharedSecre t&SharedSecretAuthServerTy pe::=HTTPBasicAuthenticati on&LoginURL::=%totpRegistr ationEndpoint%/oaa/rui/ totpPreferences/v1` **Note:** If the value of `totpRegistrationEndpoint` is not provided, then it's value is computed based on the kubernetes cluster/pod setup.	
`database.cache.type.enum.f actor.expiryTime`	Value must be greater than equal to `bharosa.uio.default.challe nge.type.enum.ChallengeOMA TOTP.registration.otpexpir ytimeMs.` If not specified default value is `600` seconds.	Specifies the cache timeout in seconds.
`bharosa.uio.default.challe nge.type.enum.ChallengeSMS .challengeCounterExpiryTim e`	`1800000`	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.fa ctor.challengecounter.expi ryTime` is used (default is 1800000 milliseconds).
`bharosa.uio.default.challe nge.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.fa ctor.retry.count` is used (default is 10).
`bharosa.uio.default.challe nge.type.enum.ChallengeSMS .appName`	`OAA`	Name of the application.
`bharosa.uio.default.challe nge.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.challe nge.type.enum.ChallengeSMS .fromAddress`	`oaa@oracle.com`	Mobile number of the SMS sending entity.

Property Name	Default Value	Description
`bharosa.uio.default.challe nge.type.enum.ChallengeSMS .fromName`	OAA	Name of the From SMS sending entity.
`bharosa.uio.default.challe nge.type.enum.ChallengeSMS .msgIPTemplate`	`IP Address:`	Part of the SMS template to display message IP address.
`bharosa.uio.default.challe nge.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.challe nge.type.enum.ChallengeSMS .msgResourceURLTemplate`	`Resource URL Access:`	Part of the SMS template to display message resource URL.
`bharosa.uio.default.challe nge.type.enum.ChallengeSMS .msgSubject`	`One Time Pin: OAA`	Subject title of the SMS template.
`bharosa.uio.default.challe nge.type.enum.ChallengeSMS .msgTimeTemplate`	`Time of Access:`	Part of the SMS template to display message time.
`bharosa.uio.default.challe nge.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.challe nge.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.challe nge.type.enum.ChallengeTOT P.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.challe nge.type.enum.ChallengeYub icoOTP.challengeCounterExp iryTime`	1800000	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.fa ctor.challengecounter.expi ryTime` is used (default is 1800000 milliseconds).
`bharosa.uio.default.challe nge.type.enum.ChallengeYub icoOTP.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.fa ctor.retry.count` is used (default is 10).

Property Name	Default Value	Description
`bharosa.uio.default.challe nge.type.enum.ChallengeFID O2.challengeCounterExpiryT ime`	1800000	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.fa ctor.challengecounter.expi ryTime` is used (default is 1800000 milliseconds).
`bharosa.uio.default.challe nge.type.enum.ChallengeFID O2.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.fa ctor.retry.count` is used (default is 10).
`oracle.security.oaa.kba.ch allenge.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.ch allenge.separator` property described in the row below.

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.
`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.

Property Name	Default Value	Description
`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.posta uth.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.<Assuran ceLevelKey>.riskcheckpoint`		Checkpoint associated with the existing assurance level. **Note:** This property is related to OAA-OARM integration.

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

Property Name	Default Value	Description
`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: ```\n[\n {\n "name":\n"rule.action.enum.Block.\npriority",\n "value": "max",\n "source":\n"database"\n }\n]\n``` **Note:** This property is related to OAA-OARM integration.
`default.all.factor.bypassChallenge.durationInMinutes`		Specifies the duration for which the user is no longer challenged after a successful login. **Note:** You can set the property to a negative value to disable this feature.

Configuration Properties for OUA

All the properties below, except `bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.retrycount` (which uses the `<PolicyUrl>` endpoint), should be set using the `<DRSS>/oaa-drss/oua/property/v1` REST API endpoint.

> **Note:**
>
> For details on the `<DRSS>` endpoint and the username and password, see Printing Deployment Details.

Property Name	Default Value	Description
General Parameters		
`echo.elapsed.time`	2	This property is required to determine the count for an unreachable device. The default value '2' means that if a device does not send an echo/heartbeat for 2 hours it will be recognized as an unreachable device.
`oua.drss.lcm.heartbeatFrequency`	1800000	Specifies the time frequency between device heartbeat calls in milliseconds.

Property Name	Default Value	Description
`oua.drss.lcm.pollingFrequency`	`43200000`	Specifies the time frequency between checking for new Oracle Universal Authenticator client software versions in milliseconds.
`oua.drss.lcm.monitoringFrequency`	`10000`	Specifies the time frequency in milliseconds used by the monitoring agent to check and restart (if required) OUADesktopHelper and OUAUpgradeAgent processes.
`oua.drss.ssoLoginUrl`		Specifies the value of the OAM endpoint. By default this value is not set and should only be set if the OAM login URL is different to the value specified for `oua.oamRuntimeEndpoint` in the `installOAA.properties`. See, Oracle Universal Authenticator Configuration. A sample value is `<http(s)>://<loginurl_host>:<Port>`.
`oua.drss.cookieParameter`	`path=/; secure; HttpOnly`	Specifies the cookie parameters for the OAM_ID cookie set by the OUA client and SSO Browser extension. This value can be changed based on your organization's security and privacy policies. For example, `secure;partitioned`. See Set-Cookie for options.
`bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.retrycount`	`10`	Specifies the maximum number of unsuccessful retries of the challenge. Beyond this count the challenge is locked. This value must be set to 50 if using OMA Push Notification Challenge with Oracle Universal Authenticator.
Password Management Parameters		
`oua.drss.password.reset.forgoturl`		Specifies the URL where users can initiate the "Forgot Password" process. Through this URL, users will be guided to reset their password by answering security questions or utilizing other recovery mechanisms configured for their account. See Password Management.
`oua.drss.password.reset.url`		Defines the URL where users can reset their password. Accessing this URL allows users to verify their identity, through such actions as answering security questions, and proceed to create a new password for their account. See Password Management.

Property Name	Default Value	Description
`oua.drss.password.reset.support edBrowsers`	chrome, firefox	Outlines the browsers supported by the system. When a forgot URL or reset URL is called from within OUA, a browser is opened. **✎ Note:** In this release, only Google Chrome and Mozilla Firefox are supported. If both browsers are installed, the system will prioritize using Chrome for optimal functionality. These browsers are required for the proper execution of this feature. See Password Management.
Configurable Challenges Parameters		
`oua.drss.skipPrimaryAuthDuratio nWithLastFullAuth`	1800 seconds (30 minutes)	Specifies the time duration from the last full OAM login. If the last full OAM login is within this time duration, the user will not be prompted for their OAM password, and will be allowed to authenticate using only the second factor. Once the duration elapses, the user will be prompted to enter their full OAM credentials, followed by a second factor.
`oua.drss.skipPrimaryAuthDuratio nWithLastMFAOnlyAuth`	600 seconds (10 minutes)	Specifies the time duration from the last successful second factor only login time. If the user performed a second factor only login within this time duration, the user will not be prompted for their OAM password, and will be allowed to authenticate using only the second factor. When their duration elapses, the user will be prompted for their OAM credentials, followed by a second factor.

Property Name	Default Value	Description
`oua.drss.skipPrimaryAuthFactorTrustLevel`	3	Specifies the trust level value for skip password rule evaluation.
		The trust level determines which factors are allowed to perform a passwordless login within the `oua.drss.skipPrimaryAuthDurationWithLastFullAuth` and `oua.drss.skipPrimaryAuthDurationWithLastMFAOnlyAuth` time periods.
		The default trust levels are as follows: • Trust Level 1 = SMS Challenge • Trust Level 2 = Yubico Yubikey TOTP, OMA TOTP • Trust Level 3 = Email Challenge • Trust Level 4 = Push Notification Challenge
		For example, if TrustLevel=3, then all those factor assigned level 3 or higher are allowed to perform passwordless login.
		Administrators can change the trust level for individual factors using the `bharosa.uio.default.challenge.type.enum.{FACTOR_KEY}.oua.trustLevel` parameters outlined in the rows below.
		✎ **Note:** FIDO2 and Security Question challenge is not currently supported with Oracle Universal Authenticator.
`bharosa.uio.default.challenge.type.enum.ChallengeSMS.oua.trustLevel`	1	Sets the trust level for the SMS Challenge.
`bharosa.uio.default.challenge.type.enum.ChallengeOMATOTP.oua.trustLevel`	2	Sets the trust level for the OMA TOTP Challenge.
`bharosa.uio.default.challenge.type.enum.ChallengeYubicoOTP.oua.trustLevel`	2	Sets the trust level for the Yubikey Yubico OTP Challenge.

ORACLE

Property Name	Default Value	Description
`bharosa.uio.default.challenge.type.enum.ChallengeEmail.oua.trustLevel`	3	Sets the trust level for the Email Challenge.
`bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.oua.trustLevel`	4	Sets the trust level for the OMA Push Challenge.
`oua.drss.allowPrimaryAuthDuringMFAOnly`	true	Determines whether the user is given the option to login with their OAM password during a second factor only login.

Configuration Properties For Customizing the User Interfaces

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

Configuration Properties For Factor Verification

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

11.10 Configuring Factor Verification

OAA allows you to configure factor verification. Factor verification allows users to verify a factor in the Self-Service Portal after the factor has been added. This allows a user to 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

11.10.1 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.

11.10.2 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, OMA Push Notification Challenge, Email Challenge, Yubico OTP Challenge, and SMS Challenge.

4. Click **Save**.

Next steps: Configuring Properties for Factor Verification.

11.10.3 Configuring Properties for Factor Verification

To enable factor verification you must set configuration properties.

Factor Verification Properties

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

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.

11.10.4 Testing Factor Verification

To test factor verification:

1. Access the Self-Service Portal 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.

11.11 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

11.11.1 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.

11.11.2 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."

11.11.3 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 11-1   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

11.11.3.1 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

11.11.3.1.1 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

11.11.3.1.2 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

11.11.3.1.3 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

11.11.3.2 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**.

11.11.4 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.

11.11.5 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.

11.11.6 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.

Disable Question or Category	New Customers	User with Disabled Question in their Question Set	Users with Question Registered
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.

11.11.7 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.

11.11.8 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.

11.12 Configuring Push Notification for Oracle Mobile Authenticator

OAA allows you to configure push notifications for the Oracle Mobile Authenticator (OMA) application.

Topics

- Configuring Oracle Mobile Authenticator Push Notification for Android
- Configuring Oracle Mobile Authenticator Push Notification for iOS

11.12.1 Configuring Oracle Mobile Authenticator Push Notification for Android

The Oracle Mobile Authenticator (OMA) is a mobile device application that uses Time-Based One-Time Passcode (TOTP) or push notifications to authenticate users with a two-factor authentication scheme. OAA allows you to configure push notifications for the OMA application.

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 application, which then communicates with the OAA server to grant or deny you access to the protected resource.

> ✏️ **Note:**
>
> Push notifications require factor verification to be configured as a prerequisite. Make sure factor verification is configured before proceeding. See Configuring Factor Verification.

Topics

The following topics describe how to configure push notification on Android:

- Installing the Oracle Mobile Authenticator Application
- Configuring Firebase and OAA
- Registering the User Account with Oracle Mobile Authenticator for Android
- Accessing a Protected Application Using Android Push Notification

11.12.1.1 Installing the Oracle Mobile Authenticator Application

You can download the Oracle Mobile Authenticator (OMA) application for Android devices from the Google Play Store.

Any end user who is to use OMA Push Notifications must install the application on their Android device.

11.12.1.2 Configuring Firebase and OAA

The following configuration steps show how to configure push notifications for Android devices with OAA.

> **Note:**
>
> Administrators should be aware of the following:
>
> - Google is deprecating Legacy FCM API's in June 2024 and migrating to HTTP v1 API's. For all new configurations it is recommended to use HTTP v1 API's. The steps in this section configure push notifications using HTTP v1 API's.
>
> - In order to use HTTPv1 API's you must be using the OAA June 24 refresh release or later.
>
> - If you have configured push notifications for Android in releases prior to OAA June 24 refresh, you will be using Legacy FCM API's. Administrators should migrate to HTTP v1 API's by upgrading to OAA June 24 refresh or later. The steps to upgrade and migrate to HTTP v1 API's can be found in Upgrading OAA, OARM, and OUA.
>
> - For reference purposes, the configuration steps using Legacy FCM API's have been moved to Configuring OMA Push Notifications Using Legacy FCM API's.

Topics:

- Creating a Google Firebase Project Enabled for Google Cloud Messaging
- Configuring OAA Properties for Android Push Notification
- Copying the Service Account JSON File to OAA

11.12.1.2.1 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 **Add project**.
3. In the **Project name** field, enter a name for your project. For example, `OAAAndroidPUSH`.
4. On the Google Analytics for your Firebase project page, deselect **Enable Google Analytics for this project**, and then click **Create project**.
5. Click **Continue** when your new project is ready.
6. In the left navigation pane of the project window, click the **Settings** icon, and then select **Project settings**.
7. On the **Project settings** page, click **Cloud Messaging**.
8. Under **Firebase Cloud Messaging API (V1)**, click on the ellipsis and select **Manage the API in Google Cloud Console**.

9. In the new tab that appears, under **Cloud Messaging** click **ENABLE** if not already enabled.

10. Return to the original tab where you clicked **Firebase Cloud Messaging API (V1)** and refresh the page.

11. Note the value present in the **Sender ID** field. This value is required later in Configuring OAA Properties for Android Push Notification.

12. Go to the **Service Accounts** tab and click **Generate new private key**. In the **Generate new private key** window, click **Generate key**. This will generate a service account json file. Download and save the file, for example `service-account.json`. Keep the file secure as it is required later in Copying the Service Account JSON File to OAA.

11.12.1.2.2 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 11-2   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.pushPreferencesEndpoint	The host and port used for push factor registration. This host and port should be accessible from the device. This corresponds to the host and port referenced in the **SpuiUrl** (`SpuiUrl=https://<host:port>/oaa/rui`) in Printing Deployment Details.	https://oaainstall
bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.challengeAnswerEndpoint	The host and port used for push factor runtime. This host and port should be accessible from the device. This corresponds to the host and port referenced in the **Push URL** (`Push=https://<host:port>/oaa-push-factor`) in Printing Deployment Details.	https://oaainstall

Table 11-2   (Cont.) OAA Properties

Property Name	Description	Sample Value
bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.retrycount	Maximum number of unsuccessful retries of the challenge. Beyond this count the challenge is locked. The default value is 10. If you are using push notifications with Oracle Universal Authenticator you must set this value to 50.	50
bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.google.firebase.senderId	The Firebase **Sender ID**. See Creating a Google Firebase Project Enabled for Google Cloud Messaging.	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.pushPreferencesEndpo
int",
"value": "https://oaainstall"
},
{
"name":
"bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.challengeAnswerEndpo
int",
"value": "https://oaainstall"
},
{
"name": "bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.retrycount",
"value": "50"
},
{
"name":
"bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.google.firebase.send
erId",
"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.

11.12.1.2.3 Copying the Service Account JSON File to OAA

This section provides information on how to store the service account json file in OAA.

The service account json file downloaded in Creating a Google Firebase Project Enabled for Google Cloud Messaging must be copied 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/gcm
   $ cp service-account.json <NFS_VAULT_PATH>/ChallengeOMAPUSH/gcm
   $ sudo chmod 444 <NFS_VAULT_PATH>/ChallengeOMAPUSH/gcm/service-account.json
   ```

2. Edit the `<NFS_CONFIG_PATH>/installOAA.properties` file and update the `common.deployment.push.gcmjsonfile` as follows and save the file:

   ```
   common.deployment.push.gcmjsonfile=/u01/oracle/service/store/oaa/
   ChallengeOMAPUSH/gcm/service-account.json
   ```

> **Note:**
>
> `/u01/oracle/service/store/oaa/ChallengeOMAPUSH/gcm/service-account.json` is the internal mapping to `<NFS_VAULT_PATH>/ChallengeOMAPUSH/gcm/service-account.json`.

3. Edit the `<NFS_LOG_PATH>/status.info` and set the following vault parameters to false and save the file:

```
VAULTINSTALL=false
VAULTCHECK=false
```

4. Enter the OAA management container, 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 /]$
```

5. Inside the OAA management pod bash shell, run the OAA.sh script to pick up the `common.deployment.push.gcmjsonfile` configuration:

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

6. Once the update to the deployment is successful, see Registering the User Account with Oracle Mobile Authenticator for Android.

11.12.1.3 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:

> **Note:**
>
> The steps below can also be found in Configuring Push Notification Challenge with Oracle Mobile Authenticator in the Oracle Advanced Authentication Self-Service Portal.

1. Log in to the Self-Service Portal 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 **Self-Service Portal** screen as the user name is case sensitive.

   b. In the **PIN code** field, enter the PIN code displayed on the **Self-Service Portal** screen.

7. Click **Sign in** and accept the certificate if prompted.

The account is successfully added in OMA.

8. On the **Self-Service Portal** screen, click **Done**.

The OMA Push Notification Challenge for the registered device appears in the Self-Service Portal.
Next Steps: Accessing a Protected Application Using Android Push Notification.

11.12.1.4 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.

11.12.2 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 Self-Service Portal) 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.

> **✎ Note:**
>
> Push notifications require factor verification to be configured as a prerequisite. Make sure factor verification is configured before proceeding. See Configuring Factor Verification.

Topics

The following topics describe how to configure push notification on iOS:

- Creating an Apple iOS Certificate, App ID, Bundle Identifier, and Keystore
- Copying the APNS Java Key Store to OAA
- Configuring OAA Properties for iOS Push Notification
- Registering the User Account with Oracle Mobile Authenticator for iOS
- Installing the Oracle Mobile Authenticator
- Accessing a Protected Application Using iOS Push Notification

11.12.2.1 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.

11.12.2.2 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.

11.12.2.3 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 11-3   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.pushPreferencesEndpoint	The host and port used for push factor registration. This host and port should be accessible from the device. This corresponds to the host and port referenced in the SpuiUrl (`SpuiUrl=https://<host:port>/oaa/rui`) in Printing Deployment Details.	https://oaainstall
bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.challengeAnswerEndpoint	The host and port used for push factor runtime. This host and port should be accessible from the device. This corresponds to the host and port referenced in the Push URL (`Push=https://<host:port>/oaa-push-factor`) in Printing Deployment Details.	https://oaainstall
bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.retrycount	Maximum number of unsuccessful retries of the challenge. Beyond this count the challenge is locked. The default value is 10. If you are using push notifications with Oracle Universal Authenticator you must set this value to 50.	50

Table 11-3   (Cont.) OAA Properties

Property Name	Description	Sample Value
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.pushPreferencesEndpo
int",
"value": "https://oaainstall"
},
{
"name":
"bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.challengeAnswerEndpo
int",
"value": "https://oaainstall"
},
{
"name": "bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.retrycount",
"value": "50"
},
{
"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.

11.12.2.4 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 Self-Service Portal 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 **Self-Service Portal** screen as the user name is case sensitive.

      b. In the **PIN code** field, enter the PIN code displayed on the **Self-Service Portal** screen.

7. Click **Sign in** and accept the certificate if prompted.

The account is successfully added in OMA.

8. On the **Self-Service Portal** screen, click **Done**.

The OMA Push Notification Challenge for the registered device appears in the Self-Service Portal.

11.12.2.5 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.

11.12.2.6 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.

11.13 Configuring OAuth JWT for REST APIs

OAA is configured to use API Key Security for REST API's by default. Administrators can enable OAuth JSON Web Tokens (JWT) to access REST APIs, which provides improved security using short-lived access tokens.

> **✎ Note:**
>
> Administrators should use **API Key Security only** mode during initial installation. Once the installation is complete and all the use-cases are verified, if you require OAuth JWT security, you should upgrade to **API Key Security and OAuth combined** mode, and verify the use-cases again. If all the use-cases are successful then, if required, you can upgrade to use **OAuth only**.

ORACLE

REST API's are used in OAA in the following circumstances:

- Internal communication between OAA pods and services.

- Performing tasks in the OAA Administration Console.

- External applications that use REST API's to communicate with OAA. The REST API's available are:

  – OAA Admin API

  – OAA Runtime API

  – OARM Risk Service API

  – OARM Customer Care Service API

Administrators can configure OAA to use one of the following modes for REST API's:

- **API Key Security only**

- **API Key Security and OAuth combined**

- **OAuth only**

> **Note:**
>
> If you are using OAA with Oracle Universal Authenticator (OUA) and you want to use OAuth JWT, you must use **API Key Security and OAuth combined** mode.

API Key Security Only

API Key Security mode is the default configuration and should be used during initial installation. When API Key Security mode is configured, internal communication, OAA Administration console tasks, and external REST API access is performed using API Key Security.

To configure API Key Security only mode, see Configuring API Key Security Only.

When accessing a REST API using an external application, the username and password is passed using Basic Authorization. The example below shows using the OAA Admin API to perform a GET request:

```
curl -i -X GET -H Authorization:Basic <Base64Encoded(<username>:<password>)> -H <request-header>:<value> <PolicyUrl>/<resource-path>
```

API Key Security and OAuth combined

When both API Key Security and OAuth is configured, the following is true:

- Internal communication and OAA Administration console tasks uses API Key Security.

- External REST API access is available using either API Key Security or OAuth JWT.

To configure API Key Security and OAuth combined mode, see Configuring API Key Security and OAuth combined.

When accessing a REST API using an external application, you can pass an access token for OAuth using `Authorization:Bearer <Token>`, or for API Key Security pass the username and password using `Authorization:Basic <Base64Encoded(<username>:<password>)>`.

OAuth Only

When OAuth only mode is used, internal OAA communication, OAA Administration console tasks, and external REST API access is performed using OAuth JWT tokens only.

To configure OAuth only mode, see Configuring OAuth Only.

When accessing a REST API using an external application, the access token for OAuth is passed using `Authorization:Bearer <Token>`. The example below shows using the OAA Admin API to perform a GET request:

```
curl -i -X GET -H Authorization:Bearer <Token> -H <request-header>:<value>
<PolicyUrl>/<resource-path>
```

11.13.1 Configuring API Key Security Only

The following steps show how to configure API Key Security only:

> **Note:**
>
> This is the default installation method configured in the `installOAA.properties`.

1. Set the following parameters under **##3. OAUTH configuration##** in the `installOAA.properties` file:

    ```
    install.global.service.security.basic.enabled=true
    install.global.service.security.oauth.enabled=false
    ```

    For more information on the `installOAA.properties` file, see Preparing the Properties file for Installation.

11.13.2 Configuring API Key Security and OAuth combined

The following sections show how to configure API Key Security and OAuth combined mode.

Preparing the installOAA.properties File

1. Set the following parameters under **##3. OAUTH configuration##** in the `installOAA.properties` file:

    ```
    install.global.service.security.basic.enabled=true
    install.global.service.security.oauth.enabled=true
    ```

    For more information on the `installOAA.properties` file, see Preparing the Properties file for Installation.

Updating OAA Using OAA.sh

After updating the `installOAA.properties` file you must run `OAA.sh` to update OAA with the configuration:

1. Connect to the 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 /]$
```

2. Inside the OAA management pod run the following to update OAA with the configuration:

```
[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.

11.13.3 Configuring OAuth Only

The following sections show how to configure OAuth only mode.

Preparing the installOAA.properties File

1. Set the following parameters under **##3. OAUTH configuration##** in the `installOAA.properties` file:

```
install.global.service.security.basic.enabled=false
install.global.service.security.oauth.enabled=true
oauth.tokenexpiry=3600
api.oauth.tokenexpiry=3600
oauth.adminname=<adminuser>
oauth.adminpassword=<adminuserpwd>
oauth.appusername=<appuser>
oauth.appuserpassword=<appuserpwd>
```

where:

- `oauth.tokenexpiry` is the expiry time for the UI token in seconds. The UI token is used when tasks are performed using the OAA Administration Console. The default value is 3600 seconds (1 hour).

- `api.oauth.tokenexpiry` is the expiry time for the API OAuth token in seconds. The API OAuth token is used for internal communications. The default value is 3600 seconds (1 hour).

- `oauth.adminname` should be set to an Administration user that is a member of the `OAA-Admin-Role` group. For more information, see Creating Users and Groups in the LDAP Store.

- `oauth.adminpassword` is the base64 encoded password for the user defined in `oauth.adminname`.

- `oauth.appusername` should be set any user that is a member of the `OAA-App-User` group. For more information, see Creating Users and Groups in the LDAP Store.

> **✎ Note:**
>
> This user can be the same user as `oauth.adminname` as long as that user exists in both the `OAA-Admin-Role` and `OAA-App-User` groups.

- `oauth.appuserpassword` is the base64 encoded password for the user defined in `oauth.appusername`.

For example:

```
install.global.service.security.basic.enabled=false
install.global.service.security.oauth.enabled=true
oauth.tokenexpiry=3600
api.oauth.tokenexpiry=3600
oauth.adminname=oaaadmin
oauth.adminpassword=bXlvYWFhZG1pbnB3ZA==
oauth.appusername=testuser
oauth.appuserpassword=bXl0ZXN0dXNlcnB3ZA==
```

For more information on the `installOAA.properties` file, see Preparing the Properties file for Installation.

Updating OAA Using OAA.sh

After updating the `installOAA.properties` file you must run `OAA.sh` to update OAA with the configuration:

1. Connect to the 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 /]$
```

2. Inside the OAA management pod run the following to update OAA with the OAuth JWT configuration:

```
[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.

Configuring the Token Refresh Cronjob

OAuth JWT access tokens are short-lived and need to be renewed prior to expiration. OAA provides a Kubernetes cronjob which automates token refresh from OAM. If preferred, Administrators can write their own automation instead of using this cronjob. To enable the cronjob perform the following steps:

1. Inside the OAA management pod run the following script to create a cronjob to renew the tokens automatically:

```
[oracle@oaamgmt-oaa-mgmt-6f4c9cd56f-std6l /]$ cd ~/scripts
[oracle@oaamgmt-oaa-mgmt-6f4c9cd56f-std6l ~]$ ./tokenRefresh.sh -f
{CONFIG_DIR}/installOAA.properties -c "<Your_Cronjob_Schedule>"
```

where "`<Your_Cronjob_Schedule>` is in the format defined at CronJob.
For example, the cronjob schedule below will renew tokens every 45 minutes:

```
./tokenRefresh.sh -f /u01/oracle/scripts/settings/installOAA.properties -c
"*/45 * * * *"
```

> **✎ Note:**
>
> You should set your token renewal interval such that tokens are renewed well in advance of `api.oauth.tokenexpiry` to ensure no downtime.

To view the status of the cronjob run:

```
kubectl get cronjob -n oaans
```

The output will look similar to the following:

```
NAME           SCHEDULE       SUSPEND   ACTIVE   LAST SCHEDULE    AGE
oaamgmt-oaa-mgmt-r   */45 * * * *   False     0           3m48s            1h
```

> **✎ Note:**
>
> Tokens that need to be retired should be revoked by invoking the appropriate OAM API. For more details, see Revoke Tokens REST Endpoints.

Troubleshooting the Token Refresh Cronjob

- The cronjob will fail under the following product conditions:
  - OAuth is not enabled in OAA, for example
    `install.global.service.security.oauth.enabled=false`
  - The job is not able get the token from the OAM Server

  The above conditions will result in cronjob pods moving to **Error** status. If the cronjob pod errors, the job is retried a configurable number of times with increasing back-off intervals. The retry attempts are stopped once `backoffLimit` is reached. When the jobs have

stopped due to errors, deleting the failed jobs will restart the cronjob. See Pod backoff failure policy .

- To view the status of the cronjob jobs and pods run the following commands:

  1. Inside the management container get the name of the cronjob:

     ```
     kubectl get cronjob -n oaans
     ```

     The output will look similar to the following:

     ```
     NAME            SCHEDULE        SUSPEND   ACTIVE   LAST SCHEDULE    AGE
     oaamgmt-oaa-mgmt-r   */45 * * * *    False     0            3m48s
     1h
     ```

  2. Run the following command to get the list of jobs associated with the cronjob :

     ```
     kubectl get jobs -l cronjob=oaamgmt-oaa-mgmt-r
     ```

     The output will look similar to the following:

     ```
     NAME                            COMPLETIONS   DURATION   AGE
     oaamgmt-oaa-mgmt-r-28646612    1/1           5s         3m9s
     ```

  3. Run the following command to get the name of the pods associated with the job:

     ```
     kubectl get pods -l cronjob=oaamgmt-oaa-mgmt-r
     ```

     The output will look similar to the following if the job pod(s) completed successfully:

     ```
     NAME                                READY   STATUS      RESTARTS   AGE
     oaamgmt-oaa-mgmt-r-28646620-brnk5   0/1     Completed   0          2m42s
     ```

     If there are problems with the job pod(s) you will see an `Error` status, for example:

     ```
     NAME                                READY   STATUS      RESTARTS   AGE
     oaamgmt-oaa-mgmt-r-28646620-brnk5   0/1     Error       0          2m42s
     oaamgmt-oaa-mgmt-r-28646620-5495n   0/1     Error       0          15s
     oaamgmt-oaa-mgmt-r-28646620-ftndp   0/1     Error       0          3s
     ```

  4. If the pod(s) status is `Error` you can diagnose the problem by viewing the logs of the most recent pod that ran. For example:

     ```
     kubectl logs oaamgmt-oaa-mgmt-r-28646620-ftndp
     ```

Accessing REST API's Using OAuth JWT from OAM

The instructions below relate only to external applications that use REST API's to communicate with OAA.

> **Note:**
>
> Internal communications and tasks performed in the OAA Administration console use the tokens configured in OAA and are refreshed by the cronjob.

1. In order to use REST API's with OAuth JWT, you need to get an access token from OAM using either a 2-legged or 3-legged flow. See Runtime REST APIs for OAuth 12c.

2. This access token should be passed using `Authorization:Bearer <Token>`. The example below shows using the OAA Admin API to perform a GET request:

   ```
   curl -i -X GET -H Authorization:Bearer <Token> -H <request-header>:<value>
   <PolicyUrl>/<resource-path>
   ```

11.14 Certificate Management and Expiry

This section explains how to manage SSL certificates that are due to expire.

The certificates used by the OAA installation are stored in PKCS files in the *<NFS_CREDS_PATH>*. These files are:

• `cert.p12` - contains the server certificate and key

• `trust.p12` - contains the trusted certificate authority certificate(s)

These files were generated either by yourself if using a commercial certificate authority, or by the installation if using self-signed certificates. See, Generating Server Certificates and Trusted Certificates.

> **Note:**
>
> Administrators should be aware of the following:
>
> • Self-signed certificates are only generated automatically by installations performed using the January 25 release or later. If you installed prior to January 25 you will have generated the self signed certificates, and `cert.p12` and `trust.p12` manually.
>
> • Self-signed certificates generated by the installation, using January 25 release or later, have a validity of 6 months.

The `trust.p12` also contains the certificates used by OAM. This includes the server certificate used by OAM, and it's full chain of trust. If you installed OAA from January 25 onwards, the original installation retrieved these certificates for you and imported them into the `trust.p12`. Prior to January 25 this was manual task and Administrators had to import them prior to installation.

Administrators should be aware of the following:

• If you are renewing third party certificates that were originally generated following Using a Third Party CA for Generating Certificates, you must recreate the `cert.p12` and `trust.p12` with the new certificate. The same instructions can be followed but you must rename your existing files before proceeding.

- If you were using self-signed certificates and now want to generate your own third party certificates, you must follow Using a Third Party CA for Generating Certificates to create a `cert.p12` and `trust.p12`.

- If you are renewing self-signed certificates, the `cert.p12` and `trust.p12` will be automatically regenerated with new certificates during the deployment update.

- If you are renewing OAM certificates, the certificate must first be renewed on the OAM server. During the deployment update, the OAM certificates will be retrieved automatically and added to the `trust.p12`.

Updating the Deployment

To update the deployment with new certificates:

1. Enter the management pod, for example:

    ```
    kubectl exec -n oaans -ti oaamgmt-oaa-mgmt-84955fdf8f-x22k4 -- /bin/bash
    ```

2. If using third party certificates and have generated your own `cert.p12` and `trust.p12` files, you must do the following:

    a. Navigate to the `/u01/oracle/scripts/creds` directory and rename the existing files:

    ```
    cd /u01/oracle/scripts/creds
    mv trust.p12 trust.p12.old
    mv cert.p12 cert.p12.old
    ```

    b. Outside the container, on the *<INSTALL_HOST>*, navigate to the *<WORKDIR>* where the new `trust.p12` and `cert.p12` files are located. For example:

    ```
    cd /workdir
    ```

    c. Run the following commands to copy the new files to the `/u01/oracle/scripts/creds/` which maps to the *<NFS_CREDS_PATH>*:

    ```
    kubectl cp <WORKDIR>/cert.p12 <NAMESPACE>/<OAAMGMTPOD>:/u01/oracle/
    scripts/creds/cert.p12
    kubectl cp <WORKDIR>/trust.p12 <NAMESPACE>/<OAAMGMTPOD>:/u01/oracle/
    scripts/creds/trust.p12
    ```

    For example:

    ```
    kubectl cp /workdir/cert.p12 oaans/oaamgmt-oaa-mgmt-84955fdf8f-
    x22k4:/u01/oracle/scripts/creds/cert.p12
    kubectl cp /workdir/trust.p12 oaans/oaamgmt-oaa-mgmt-84955fdf8f-
    x22k4:/u01/oracle/scripts/creds/trust.p12
    ```

3. Inside the management container, edit the `/u01/oracle/scripts/settings/installOAA.properties` and ensure the following parameters are set:

    > **✎ Note:**
    >
    > These parameters should be already set from the original installation, unless you are moving from self-signed certificates to third party certificates.

ORACLE

Variable	Sample Value	Description
`common.deployment.keystorepass phrase=`*`<USER_CERT_P12_PWD>`*	password	If using your own certificates, you must set this to *<USER_CERT_P12_PWD>*. If you are going to use the self-signed certificates generated by OAA during installation, then set to a password of your choice.
`common.deployment.truststorepa ssphrase=`*`<TRUST_CERT_P12_PWD>`*	password	If using your own certificates, you must set this to *<TRUST_CERT_P12_PWD>*. If you are going to use the self-signed certificates generated by OAA during installation, then set to a password of your choice.

4. Edit the `/u01/oracle/logs/status.info` and change the following two parameters to FALSE

```
CERTIFICATEINSTALL=false
VAULTINSTALL=false
```

5. Run the following command to update the deployment:

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

This command will update the deployment with the new certificates.

During the deployment update, progress is shown on the screen. If you need more detailed information, you can view the `install.log`. The log is accessible from within the management container at `/u01/oracle/logs`, or outside the container at *<NFS_LOGS_PATH>*.

If the deployment update is successful you will see output similar to Printing Deployment Details.

> **Note:**
>
> If you encounter problems with the deployment update, see Troubleshooting the Installation.

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.

If you are installing OAA December 24 or later, this is installed and configured for you. For releases prior to December 24, this must be configured manually.

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

If you installed using December 24 release or later, this is configured for you during installation. For releases prior to December 24, to configure OAM with OAA, or if you want to know more

about how this OAM and OAA integration works, 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.

Optional OIM Properties

There are some optional OIM properties that you can configure using the REST APIs.

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/`
`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.

Property	Description
`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

- Customizing Email and SMS Messaging Provider
- Customizing the OAA User Interface

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.customizedProvid
er"
"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=co
m.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.customizedProvider
"
"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, Self-Service Portal 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 keep in mind the following 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 Self-Service Portal 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.

Configuration Properties to Customize the Administration Console UI

Property Name	Description	Sample Value
oaa.admin.ui.theme.default.image.path.logo	Logo image path	https://www.example.com/content/images/logo.jpg
oaa.admin.ui.theme.default.image.path.logo.useMaxSpace	Size for custom logos. By default custom logos render to 50x42 pixels (px), regardless of size. For example, if the logo is 40x40 px, the logo will stretch to 50x42 px. If you set useMaxSpace to true, the dimensions of the 40x40px logo will remain intact. This is true as long as the logo width is < 200px and logo height is <42 px. Anything over this limit will resize to 200x42px.	true
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
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

Property Name	Description	Sample Value
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 Self-Service Portal UI

Learn about the configuration properties that you can set to customize the Self-Service Portal UI.

Configuration Properties to Customize the Self-Service Portal 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.logo.useMaxSpace	Size for custom logos. By default custom logos render to 50x42 pixels (px), regardless of size. For example, if the logo is 40x40 px, the logo will stretch to 50x42 px. If you set useMaxSpace to true, the dimensions of the 40x40px logo will remain intact. This is true as long as the logo width is < 200px and logo height is <42 px. Anything over this limit will resize to 200x42px.	true
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
oaa.prefs.ui.theme.default.footer.text.contactus	Footer text for "contactus".	Contact us

Property Name	Description	Sample Value
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 © 2025, 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 Configuration Values for 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.menu.button.text.color	To change the color of the menu button text.	rgb(31,92,255)
oaa.prefs.ui.theme.default.primary.button.color.focus	To change the color of the submit button. Deprecated from February 2025 release.	rgb(31,92,255)
oaa.prefs.ui.theme.default.primary.button.color.hover	To change the color of the submit button. Deprecated from February 2025 release.	rgb(31,92,255)
oaa.prefs.ui.theme.default.primary.button.color.active	To change the color of the submit button. Deprecated from February 2025 release.	rgb(31,92,255)
oaa.prefs.ui.theme.default.primary.button.color	To change the color of the primary button. For example, Save and Confirm buttons.	rgb(31,92,255)
oaa.prefs.ui.theme.default.primary.button.text.color	To change the color of the text of the primary button. For example, Save and Confirm buttons.	rgb(31,92,255)
oaa.prefs.ui.theme.default.secondary.button.active.color	To change the color of secondary buttons. For example,filter buttons when selected. When filter buttons are not selected they are treated as default (oaa.prefs.ui.theme.default.default.button.color).	rgb(31,92,255)

Property Name	Description	Sample Value
oaa.prefs.ui.theme.default.secondary.button.active.text.color	To change the color of the secondary button text. For example, filter buttons when selected. When filter buttons are not selected they are treated as default (oaa.prefs.ui.theme.default.default.button.text.color).	rgb(31,92,255)
oaa.prefs.ui.theme.default.tertiary.button.color	To change the color of the tertiary buttons. For example, the Manage button on the Self Service Portal home page.	rgb(31,92,255)
oaa.prefs.ui.theme.default.tertiary.button.text.color	To change the color of the tertiary button text.	rgb(31,92,255)
oaa.prefs.ui.theme.default.default.button.color	To change the color of the default buttons. For example, Cancel or filter buttons when not selected.	rgb(31,92,255)
oaa.prefs.ui.theme.default.default.button.text.color	To change the color of the default button text. For example, Cancel buttons or filter buttons when not selected.	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.

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.logo.useMaxSpace	Size for custom logos. By default custom logos render to 50x42 pixels (px), regardless of size. For example, if the logo is 40x40 px, the logo will stretch to 50x42 px. If you set useMaxSpace to true, the dimensions of the 40x40px logo will remain intact. This is true as long as the logo width is < 200px and logo height is <42 px. Anything over this limit will resize to 200x42px.	true
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

ORACLE

Property Name	Description	Sample Value
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)
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.

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
'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

ORACLE

Font family	Possible Values
'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
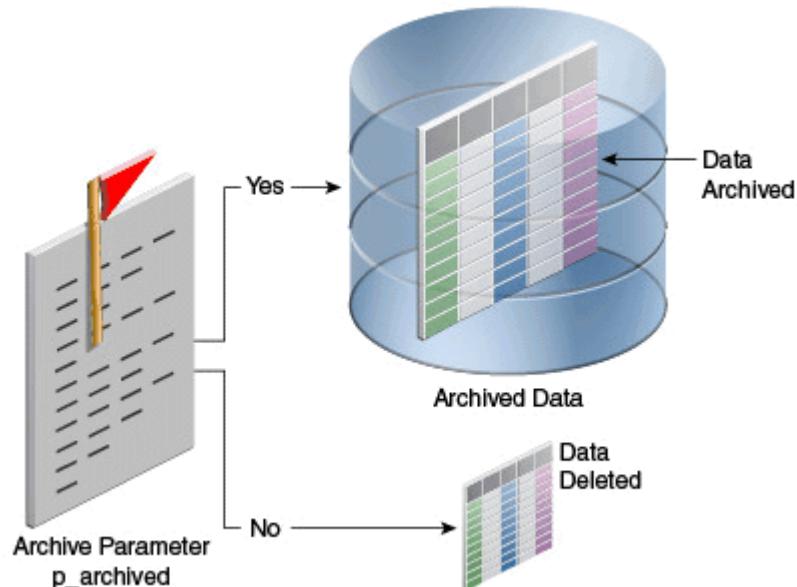
"Purging" is different from "backing up data". A data backup is for the recovery of data if loss occurs; purges are for keeping the runtime tables free of old data. Regardless, to protect your data, database backups should be performed on a regular basis with the help of a database administrator.

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
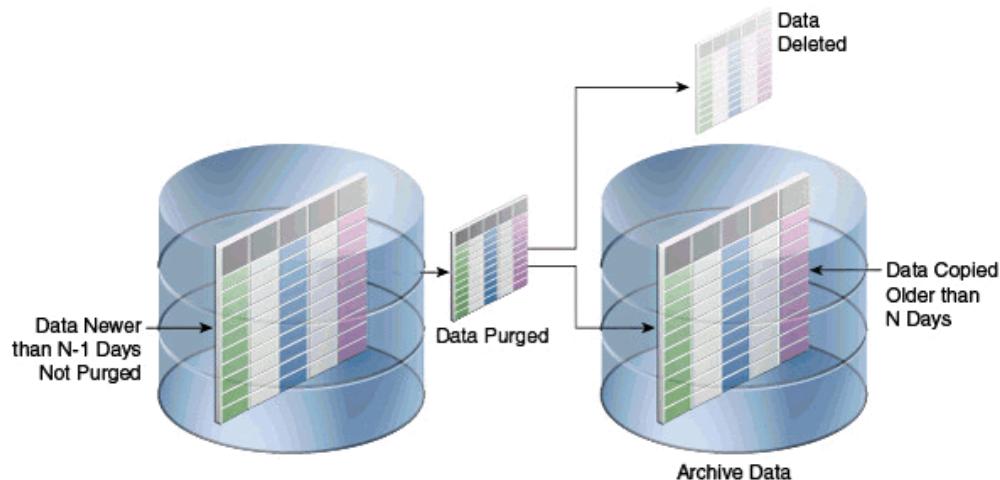
2. Determine whether to purge or archive.

Figure 14-3   Determining to Purge or Archive
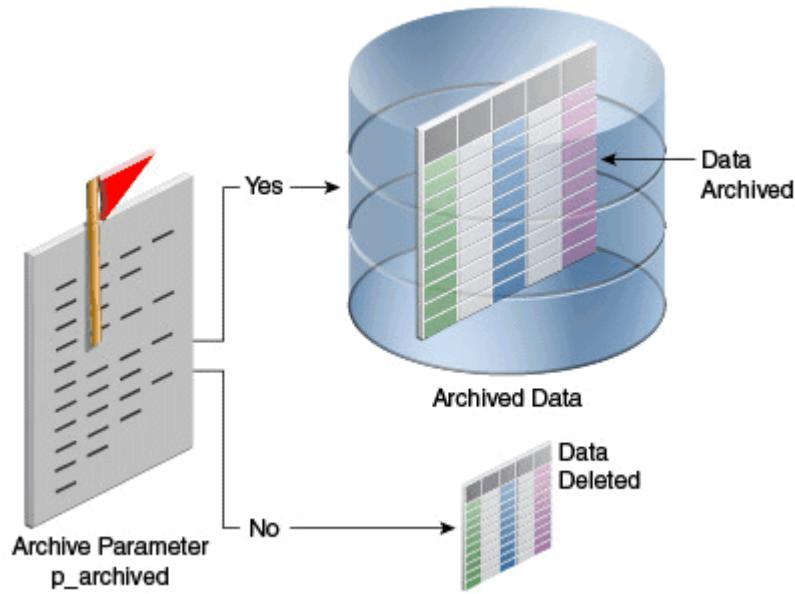
3. Deploy the purge related stored procedures into the OAA database. This is a one-time job.

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

Archive and Purge Scripts Based on Types of Data

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:

Type of Data	Corresponding Script	Description/Comments
User Authentication related data	exec_sp_purge_tracker_data.sql	All OAA/OARM/OUA 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.
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/OUA 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. In cases where the sql script prompts for `p_days1` and/or `p_archived` then there is no need to edit the sql script before running. The table below describes these 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. Run the following commant to set the `NLS_DATE_FORMAT` correctly:

```
ALTER Session SET NLS_DATE_FORMAT='YYYY-MM-DD';
```

3. 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

Minimum Data Retention Policies

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.

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.

Data	Retention Policy
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

User Authentication related Data

Details of data that is purged and the corresponding archived tables, are presented below:

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

User Authentication related Tables	Corresponding Archived Tables
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
V_DEVICE_DETAILS_ACTIVITY	V_DEVICE_DETAILS_ACTIVITY_PURGE
V_DEVICE_ASSIGNMENT_DETAILS_ACTIVITY	V_DEVICE_ASSIGNMENT_DETAILS_ACTIVITY_PURGE
V_DRSS_USER_LOGIN_DATA	V_DRSS_USER_LOGIN_DATA_PURGE
V_SOFTWARE_VERSION_DETAILS_ACTIVITY	V_SOFTWARE_VERSION_DETAILS_ACTIVITY_PURGE
V_DEVICE_SOFTWARE_VERSION_DETAILS_ACTIVITY	V_DEVICE_SOFTWARE_VERSION_DETAILS_ACTIVITY_PURGE

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

Custom Activities related Data

Transaction Tables	Corresponding Archived Tables
VT_ENTITY_ONE	VT_ENTITY_ONE_PURGE
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

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

ORACLE

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

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:

- Key Use Cases of OARM for Enhanced Security and Risk Mitigation
- Device Fingerprinting and Identification

16

Key Use Cases of OARM for Enhanced Security and Risk Mitigation

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 User Authentication activity and the rules referenced in Out-of-the-Box User Authentication Rules Supported to perform a wide range of tasks. This section describes the following example use cases:

- Configuring a Risky IP Use Case
- Configuring a Geo-Velocity Based Use Case
- Loading Geo-Location Data

See Configuring a Custom Activity Use Case in Oracle Adaptive Risk Management tutorial for detailed instructions on configuring a custom activity in OARM.

Details on how OARM processes custom user activities and provides values for API operations from the client application can be found in Understanding the Sequence of User Activity Runtime API Calls.

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 **Block based on Risky IP** out-of-the-box rule. The outcome of configuring this rule is to block 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.
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`.

   Block based on Risky IP rule appears that you need to configure for this use case.

5. Click the **Edit** icon against the **Block based on Risky IP** rule.

> **Note:**
>
> The **Block 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 **Block** 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 challenge the user, then you can configure the action as **Challenge**.
> See Configuring a Risky IP Use Case in Oracle Adaptive Risk Management tutorial for detailed instructions on using this rule in OARM.

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:

    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. Consequently, the user is blocked.

> **Note:**
>
> To learn how to configure factors, see Managing Factors in the Self-Service Portal.

ORACLE

> **✏ Note:**
>
> To learn how to create a rule effectively, see Add New Rule.

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 Self-Service Portal.

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
```

ORACLE

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.

16.4 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.

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

OS/Browser	Characteristics
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,localv
ariant,userAgent
vcrypt.fingerprint.type.enum.browser.search_list=locallang,userAgent
vcrypt.fingerprint.type.enum.browser.result_list=locallang,userAgent
vcrypt.fingerprint.type.enum.browser.header_value_nv=t,true,f,false,en,English
,es,Spanish,de,German,it,Italian,ja,Japanese,fr,French,ko,Korean,zh,Chinese,ar
,Arabic,cs,Czech,da,Danish,nl,Dutch,fi,Finnish,el,Greek,iw,Hebrew,hu,Hungarian
,no,Norwegian,pl,Polish,pt,Portuguese,ro,Romanian,ru,Russian,sk,Slovak,sv,Swed
ish,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,ua
,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,Engl
ish,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,S
wedish,th,Thai,tr,Turkish,BR,Brazil,CA,Canada
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.core.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 installOAA.properties Parameters

The following sections provide a complete list of all the parameters in the default `installOAA.properties` file.

- Common Deployment Configuration
- Database Configuration
- OAM OAuth Configuration
- Vault configuration
- Helm Chart Configuration
- Optional Configuration
- Ingress Configuration
- Management Container Configuration
- Oracle Universal Authenticator Configuration
- LDAP Configuration
- Oracle Advanced Authentication TAP Configuration

A.1 Common Deployment Configuration

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

Common Deployment Configuration

Properties	Mandatory/Optional	Installation Type	Description
`common.dryrun`	Optional	All	If enabled and set to true, the helm installation will only display generated values and will not actually perform the OAA/OARM/OUA installation on the Kubernetes cluster. This is equivalent to `--dry-run --debug` option in the helm command.

Properties	Mandatory/Optional	Installation Type	Description
common.deployment.name	Mandatory	All	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.
common.deployment.overridefile	Optional	All	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. See, Advanced Configuration with OAA Override File.
common.kube.context	Optional	All	Name of the Kubernetes context to be used. If the context is not provided, the default Kubernetes context is used.
common.kube.namespace	Optional	All	The namespace where you want to create the deployment. This should be the namespace created in Creating a Kubernetes Namespace and Secret. If the parameter is not set it will deploy to the default namespace.

Properties	Mandatory/Optional	Installation Type	Description
`common.deployment.sslcert`	Mandatory	All	The server certificate PKCS12 file to be used in the 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/OARM/OUA microservices
`common.deployment.trustcert`	Mandatory	All	The trusted certificate PKCS12 file to be used in the 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/OARM/OUA microservices
`common.deployment.importtruststore`	Mandatory	All	If this is enabled then the trusted certificate is imported in the JRE truststore.
`common.deployment.keystorepassphrase`	Mandatory	All	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.

Properties	Mandatory/Optional	Installation Type	Description
`common.deployment.t ruststorepassphrase`	Mandatory	All	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.

Properties	Mandatory/Optional	Installation Type	Description

`common.deployment.g enerate.secret`	Mandatory	All	If set to true, the installation generates three symmetric keys and adds them to the `cert.p12` referenced by the parameter `common.deployment.s slcert`. The encryption keys generated are:
			• spui-enckey - This key is used by the SPUI service for encryption.
			• aes256_db_key_ali as - 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 $keynametou se -keyalg $KEYALGO -keystore $KEYSTOR E -storepass $STOREP ASS -storetype $STORET YPE -keysize $KEYSIZE`
			
			for example:
			
			`keytool -genseckey -alias spui-enckey -`

Properties	Mandatory/Optional	Installation Type	Description
			`keyalg AES -keystore cert.p12 -storepass <password> -storetype PKCS12 -keysize 256`
`common.deployment.mode`	Mandatory	All	The following values can be set in `installOAA.properties` • `Both` - install OAA and OARM. • `OAA` - install OAA only. • `Risk` - install OARM only. • `OUA` - install OAA, OARM, and OUA.
`common.migration.configkey`	Optional	All	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	All	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.
`common.oim.integration`	Optional	All except OARM only installations	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.

Properties	Mandatory/Optional	Installation Type	Description
`common.deployment.push.apnsjksfile`	Optional	All except OARM only installations	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.
`common.deployment.push.gcmjsonfile`	Optional	All except OARM only installations	File used when enabling push notifications for Android devices. You need to set this only if you have already have created your Google Firebase project and downloaded the service account json file prior to install. Else, you can configure this post installation. The `service-account.json` file should be copied to the `<NFS_VAULT_PATH>/ChallengeOMAPUSH/gcm/` directory. The value should be set to `/u01/oracle/service/store/oaa/ChallengeOMAPUSH/gcm/service-account.json`. For more details, see Configuring Oracle Mobile Authenticator Push Notification for Android.

A.2 Database Configuration

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

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.
`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.

> **Note:**
>
> The schema name cannot exceed twelve characters and must be in uppercase.

Properties	Mandatory/Optional	Description
`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.

Configuring SSL between OAA and the Database

If required, you can configure one-way SSL between OAA and the Oracle Database. The database can use a certificate from either a commercial Certificate Authority (CA), a certificate from a private CA, or a self-signed certificate.

If the database is using a certificate issued by a private CA, or is using a self-signed certificate, then you must create an Oracle Wallet for use with OAA.

To configure OAA to use SSL communication to the database, perform the following steps:

1. If the database is using a certificate issued by a private CA, or is using a self-signed certificate, you must create an Oracle Wallet. Consult your usual Oracle Database documentation for instructions for how to perform these steps:

   > **Note:**
   >
   > If the database is using a certificate issued by a commercial CA, there is no need to create a wallet as OAA contains commercial CA's by default.

   a. If your database uses a certificate issued by a private CA, import the CA certificate(s) that signed the database certificate as a Trusted Certificate in the Wallet. If your database uses a self-signed certificate, import the self-signed certificate as a Trusted Certificate in the Wallet.

   b. Once imported, save the Wallet, and add the Wallet to a zip file, for example `wallet.zip`.

   c. Copy the `wallet.zip` file to the *<INSTALL_HOST>*.

2. Follow Creating the Management Container and run as follows:

   ```
   ./installManagementContainer.sh -E -s <path_to>/wallet.zip -t ./<oaa-
   image>.tar
   ```

> **Note:**
>
> If your database is using a certificate issued by a commercial CA you do not need to pass the `-s <path_to>/wallet.zip` parameter.

The `installManagementContainer.sh` command will automatically add two new parameters to the `installOAA.properties`:

- `database.protocol=tcps`

- `database.wallet_location` - This will only be added if `-s <path_to>/wallet.zip` is used.

3. Install OAA by following Deploying OAA, OARM, and OUA.

A.3 OAM OAuth Configuration

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

OAM OAuth Configuration

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

Properties	Mandatory/Optional	Description
`oauth.enabled`	Mandatory	OAuth is required if you want to use the Administration Console and Self-Service Portal.
		If access to the Administration Console and Self-Service Portal is required, you must set this to `true` to enable OAuth in the OAA installation.
		If you do not want access to the Administration Console and Self-Service Portal 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`
`install.global.service.security.oauth.enabled`	Optional	Controls whether to turn on OAuth for REST API calls. The default value is false and should not be changed during initial installation. Post installation, this value should not be set to true before reading and understanding Configuring OAuth JWT for REST APIs.
`install.global.service.security.basic.enabled`	Optional	Controls whether to use basic authentication for REST API calls. The default value is true. This value should not be changed during initial installation. Post installation, this value should not be set to false before reading and understanding Configuring OAuth JWT for REST APIs.
`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.

Properties	Mandatory/Optional	Description
`oauth.domainname`	Mandatory if `oauth.createdomain` is set to `true`	Specify the OAuth domain name. This must be same as the `<DomainName>` provided in Configuring OAuth and Oracle HTTP Server.
`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.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.
`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 Administration 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.`
`oauth.createclient`	Optional	Creates the OAuth client. The OAuth client is required if `oauth.enabled` is set to `true`.

Properties	Mandatory/Optional	Description
`oauth.clientname`	Mandatory if `oauth.createclient` is set to `true`	Specify the OAuth client name that will be created during the installation.
`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.tokenexpiry`	Mandatory if both the following properties are set as: `install.global.service.security.oauth.enabled=true` and `install.global.service.security.basic.enabled=false`.	UI OAuth token expiry in seconds. Default value is 3600 seconds (one hour). This should only be configured post installation. See Configuring OAuth JWT for REST APIs.
`api.oauth.tokenexpiry`	Mandatory if both the following properties are set as: `install.global.service.security.oauth.enabled=true` and `install.global.service.security.basic.enabled=false`.	API OAuth token expiry in seconds. Default value is 3600 seconds (one hour). This should only be configured post installation. See Configuring OAuth JWT for REST APIs.
`oauth.adminname`	Mandatory if both the following properties are set as: `install.global.service.security.oauth.enabled=true` and `install.global.service.security.basic.enabled=false`.	Value must be set to an Administration user that is a member of the `OAA-Admin-Role` group. This should only be configured post installation. See Configuring OAuth JWT for REST APIs.

Properties	Mandatory/Optional	Description
`oauth.adminpassword`	Mandatory if both the following properties are set as: `install.global.service.security.oauth.enabled=true` and `install.global.service.security.basic.enabled=false.`	Value must be set to the base64 password of the Administration user set for `oauth.adminname`. This should only be configured post installation. See Configuring OAuth JWT for REST APIs.
`oauth.appusername`	Mandatory if both the following properties are set as: `install.global.service.security.oauth.enabled=true` and `install.global.service.security.basic.enabled=false.`	Value must be set to any user that is a member of the `OAA-App-User` group. This should only be configured post installation. See Configuring OAuth JWT for REST APIs.
`oauth.appuserpassword`	Mandatory if both the following properties are set as: `install.global.service.security.oauth.enabled=true` and `install.global.service.security.basic.enabled=false.`	Value must be set to the base64 password of the Administration user set for `oauth.appuserpassword`. This should only be configured post installation. See Configuring OAuth JWT for REST APIs.
`oauth.provider`	Mandatory	The OAuth provider. The only valid value is `OAM`.
`oauth.serverapprole`	Optional	The LDAP group(s) that contain the users who are permitted to access the Self-Service Portal. The default value if not set is `OAA-App-User`. The following values can also be set: • `OAA-All-Role` - Users can be members of any LDAP group. • `group1,group2,group3,etc` - Users must be members of the LDAP groups listed. Groups must be provided in a comma separated list. In case of custom groups, the group names must be correct and not contain forbidden characters. If you choose to disable access to a custom group, you must adjust this property accordingly and run `OAA.sh` again to pick up the changes. See Deploying OAA, OARM, and OUA.

A.4 Vault configuration

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

Vault Configuration

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

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.
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.

Properties	Description
`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`.

A.5 Helm Chart Configuration

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

Helm Chart Configuration

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

Properties	Mandatory/Optional	Description
`install.global.repo`	Mandatory	Specify the Container Image Registry where the OAA container images exists. For more details, see Setting Up a Container Image Registry (CIR)
`install.global.testrepo`	Optional	Speficies an alternate Container Image Registry where container images can be pulled. For example, OAA installs `oraclelinux:8-slim` from an external site (`https://ghcr.io/oracle`). If your Kubernetes cluster does not have access to the internet, you must pull this image and store it in your container registry. Then you must set `install.global.testrepo` to the location of your container registry.
`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.

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 that you set earlier e.g `docker secret`. For more details, see Creating a Kubernetes Namespace and Secret.

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.template` to `installOAA.properties` this tag will be already set.
`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.

Properties	Mandatory/Optional	Description
`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`	Mandatory for any install type that installs OARM.	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, OARM only, or OAA-OARM-OUA installation.
`install.global.drssapikey`	Mandatory for OAA-OARM-OUA installations.	Specify the REST API key to be used for protecting REST endpoints in the OAA DRSS microservice. This parameter is mandatory if performing an OAA-OARM-OUA 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.

A.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.ena bled`	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.run time.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.adm in.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.dscredentia ls` `install.global.dbservicena me`	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.oidci dentityuri` `install.global.oauth.oidca udience` `install.global.oauth.oidcc lientid`	Optional	The following properties are related to OAuth. If they are not specified here, the values provided in the OAuth Configuration are used.
`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.

Properties	Mandatory/Optional	Description
`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 Admininistration console (`oaa-admin-ui`), Self-Service Portal (`spui`) , 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=Clust erIP`
`install.oaa-admin-ui.service.type=NodePort`
`install.oaa-policy.service.type=NodePo rt`
`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=N odePort`
`install.yotp.service.type= NodePort`
`install.risk.service.type= NodePort`
`install.oaa-kba.service.type=NodePort`
`install.oaa-drss.service.type=NodePort`
`install.risk.riskcc.servic e.type=NodePort`
	Optional	Default service type for services is ClusterIP.
		When deployment mode is Risk the following service are not deployed : fido, push, yotp, email ,sms, totp and kba.
		If `install.global.ingress.ena bled=true` all these parameters except `install.service.type=Clust erIP` should be commented out.

For details on installing using ingress, see: Installing NGINX Ingress Controllers

A.7 Ingress Configuration

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

Table A-1   Ingress Configuration

Properties	Mandatory/Optional	Description
`ingress.install`	Mandatory	Set value to `true` if you want the 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.ena bled=true` must also be set in Optional Configuration.

Table A-1   (Cont.) Ingress Configuration

Properties	Mandatory/Optional	Description
`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=nginx`	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.
`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 NGINX Ingress Controllers

A.8 Management Container Configuration

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

Table A-2   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>`.

Table A-2   (Cont.) Management Configuration

Properties	Mandatory/Optional	Description
`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. This only needs to be set If you have generated your own certificates as per Generating Server Certificates and Trusted Certificates. If you want to use self signed certificates, do not set this.
`common.local.trustcert`	Mandatory	Set the value to the local PATH where the trusted certificate PKCS12 file (`trust.p12`) resides. This only needs to be set If you have generated your own certificates as per Generating Server Certificates and Trusted Certificates. If you want to use self signed certificates, do not set this.

For details on NFS mounts, see: Configuring NFS Volumes

A.9 Oracle Universal Authenticator Configuration

This section provides details about the Oracle Universal Authenticator (OUA) configuration properties that can be set in the `installOAA.properties`.

Table A-3   Oracle Universal Authenticator Configuration

Properties	Mandatory/Optional	Description
oua.tapAgentName	Mandatory for OUA	The default value is `OAM-OUA-TAP`. This should not be changed.
		This should be set to the value of the TAP partner name created in section **Registering OUA as a TAP Partner in OAM** in Registering OAM TAP Partners.
oua.tapAgentFileLocation	Mandatory for OUA	Set the value to the local PATH and file name of the keystore for the OAM OAA TAP partner created in section **Registering OUA as a TAP Partner in OAM** in Registering OAM TAP Partners.
		✎ **Note:** The PATH must accessible to the OAA management container.
oua.tapAgentFilePass	Mandatory for OUA	Set the value to the Base64 encoded password of the TAP partner keystore generated in section **Registering OUA as a TAP Partner in OAM** in Registering OAM TAP Partners.
oua.oamRuntimeEndpoint	Mandatory for OUA	The OHS URL used as the entry point to OAM. If a load balancer front ends the OHS then this value is the load balancer URL.

A.10 LDAP Configuration

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

Table A-4   LDAP Configuration

Properties	Mandatory/Optional	Description
`ldap.server`	Mandatory for OAA	The LDAP server protocol, hostname and port for the LDAP server used by OAM.
`ldap.username`	Mandatory for OAA	The user name of the directory administrator.
`ldap.password`	Mandatory for OAA	The password of the directory administrator.
`ldap.oaaAdminUser`	Mandatory for OAA	The OAA administration user to be created in the LDAP user search base.
`ldap.adminRole`	Mandatory for OAA	The OAA-Admin-Role group to be created in the LDAP group search base.
`ldap.userRole`	Mandatory for OAA	The OAA-App-User group to be created in the LDAP group search base.
`ldap.oaaAdminUserPwd`	Mandatory for OAA	Set to a password of your choice. This will be the password for the `ldap.oaaAdminUser.`
`ldap.addExistingUsers`	Mandatory for OAA	Set this value to `yes` if you want the OAA installation to add all your existing users in your *<LDAP_USER_SEARCHBASE>* to the OAA-App-User group. See Creating Users and Groups in the LDAP Store for more details.

A.11 Oracle Advanced Authentication TAP Configuration

This section provides details about the Oracle Advanced Authentication TAP configuration properties that can be set in the `installOAA.properties.`

Table A-5   Oracle Advanced Authentication TAP Configuration

Properties	Mandatory/Optional	Description
`oaa.tapAgentName`	Mandatory for OAA	The default value is `OAM-OAA-TAP`. This should not be changed if doing a new installation from December 24 onwards. This should be set to the value of the TAP partner name created in section **Registering OAA as a TAP Partner in OAM** in Registering OAM TAP Partners. If performing an upgrade from a release prior to December 24, seeUpgrading OAA, OARM, and OUA before upgrading.
`oaa.tapAgentFileLocation`	Mandatory for OAA	Set the value to the local PATH and file name of the keystore for the TAP partner created in section **Registering OAA as a TAP Partner in OAM** in Registering OAM TAP Partners. ✎ **Note:** The PATH must accessible to the OAA management container.
`oaa.tapAgentFilePass`	Mandatory for OAA	Set the value to the Base64 encoded password of the TAP partner keystore generated in section **Registering OAA as a TAP Partner in OAM** in Registering OAM TAP Partners.

Table A-5   (Cont.) Oracle Advanced Authentication TAP Configuration

Properties	Mandatory/Optional	Description
`oaa.authFactors`	Mandatory for OAA	The authentication factors to be created for the OAA OAM integration agent during a new installation. The default value is `ChallengeEmail,ChallengeSMS,ChallengeOMATOTP,ChallengeYubicoOTP,ChallengeOMAPUSH,ChallengeFIDO2.`

B
Advanced Configuration with OAA Override File

The OAA Override file is used to determine the number of each type of container that is started. By default, the installation starts one pod for each container, with predefined memory and CPU requirements. For a sandbox architecture is it not usually required to change these defaults. In a highly available production deployment, there should be a minimum of two for each container type.

If you need to change the number of containers to be started, enter the management container, for example:

```
kubectl exec -n oaans -ti oaamgmt-oaa-mgmt-84955fdf8f-x22k4 -- /bin/bash
```

and update the `/u01/oracle/scripts/settings/oaaoverride.yaml` file to increase the `replicaCount` for each type of container to the required quantity.

You can also use this file to specify the CPU and memory requirements. By declaring resource requirements, you ensure that a particular OAA pod is started only on a worker node that has sufficient capacity to service a pod with these resource requirements.

The following shows an example `oaaoverride.yaml`. You can change, add, or remove parameters based on your requirements:

```
#override file for oaa installation

#if database is external to the cluster set the flag to ExternalName
riskdb:
  service:
    type: ExternalName

#replica count of oaa service
replicaCount: 2

#The following properties define the dependency spui service and can be
overridden here.
spui:
  resources:
    requests:
      cpu: 200m
      memory: "1Gi"
  replicaCount: 2

#The following properties define the dependency totp service and can be
overridden here.
totp:
  resources:
    requests:
      cpu: 200m
```

```
      memory: "1Gi"
  replicaCount: 2

#The following properties define the dependency yotp service and can be
overridden here.
yotp:
  resources:
    requests:
      cpu: 200m
      memory: "1Gi"
  replicaCount: 2

#The following properties define the dependency fido service and can be
overridden here.
fido:
  resources:
    requests:
      cpu: 200m
      memory: "1Gi"
  replicaCount: 2

#The following properties define the dependency oaa-admin-ui service and can
be overridden here.
oaa-admin-ui:
  resources:
    requests:
      cpu: 200m
      memory: "1Gi"
  replicaCount: 2

#The following properties define the dependency email service and can be
overridden here.
email:
  resources:
    requests:
      cpu: 200m
      memory: "1Gi"
  replicaCount: 2

#The following properties define the dependency push service and can be
overridden here.
push:
  resources:
    requests:
      cpu: 200m
      memory: "1Gi"
  replicaCount: 2

#The following properties define the dependency sms service and can be
overridden here.
sms:
  resources:
    requests:
      cpu: 200m
      memory: "1Gi"
  replicaCount: 2
```

ORACLE®

```
#The following properties define the dependency oaa-policy service and can be
overridden here.
oaa-policy:
  resources:
    requests:
      cpu: 200m
      memory: "1Gi"
  replicaCount: 2

#The following properties define the defaults of risk and riskcc services.
risk:
  resources:
    requests:
      cpu: 200m
      memory: "1Gi"
  replicaCount: 2

riskcc:
  resources:
    requests:
      cpu: 200m
      memory: "1Gi"
  replicaCount: 2
#
#The following properties define the defaults of customfactor service.
customfactor:
  resources:
    requests:
      cpu: 200m
      memory: "1Gi"
  replicaCount: 2
#
#The following properties define the defaults of oaa-kba service.
oaa-kba:
  resources:
    requests:
      cpu: 200m
      memory: "1Gi"
  replicaCount: 2

#
#The following properties define the defaults of oaa-drss service.
oaa-drss:
  resources:
    requests:
      cpu: 200m
      memory: "1Gi"
  replicaCount: 2
```

For the deployment to use this file, you must enter the management container, and update
the `/u01/oracle/scripts/settings/installOAA.properties` by uncommenting the following
parameter:

```
common.deployment.overridefile=/u01/oracle/scripts/settings/oaaoverride.yaml
```

To update the deployment, run the following commands:

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

C

Installing NGINX Ingress Controllers

The instructions in this section are for installing an ingress controller outside the recommended architectures in Supported Architectures.

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

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

  If you install an ingress controller as part of the 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 Installation

- **Installing your own Ingress Controller to use HTTPS**

  If you need to use install an Ingress controller on HTTPS and use your own certificates.

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

C.1 Installing Ingress Controller During OAA Installation

To install an Ingress Controller supplied with the OAA installation, you must edit the `installOAA.properties` file and update the **Optional Configuration** section with ingress 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.

Additional Considerations

When installing the Ingress Controller supplied with the OAA installation, the following additional image is installed:

- `controller:v1.0.0` from `https://registry.k8s.io/ingress-nginx`.

Administrators must whitelist this site to allow the Kubernetes cluster to pull this image.

If you cannot whitelist this site, then you must pull the image down manually and store it in your container registry.

It is recommended to pull the latest `controller:v1.X` version. See https://github.com/kubernetes/ingress-nginx/releases to find the latest release. For example:

```
podman pull registry.k8s.io/ingress-nginx/controller:v1.X.X
```

In order for the installation to know about the location of the `controller:v1.X.X` image, add the following parameters to the `installOAA.properties` in the **##7. Ingress configuration##** section:

> **Note:**
>
> These parameters are not shown in the `installOAA.properties` file by default.

```
ingress.install.controller.image.repository=<registry>
ingress.install.controller.image.image=<repository>
ingress.install.controller.image.tag=<tag>
```

For example:

```
ingress.install.controller.image.repository=container-registry.example.com
ingress.install.controller.image.image=ingress-nginx/controller
ingress.install.controller.image.tag=v1.X.X
```

See Understanding installOAA.properties Parameters.

C.2 Installing your own Ingress Controller on HTTPS

Install your own Ingress controller using SSL and your own certificates.

This section provides the steps required for installing your own ingress controller using SSL and your own certificates.

Topics

- Installing the Ingress Controller
- Updating the Install Properties File for Installing OAA Using Ingress

C.2.1 Installing the Ingress Controller

You must install the ingress controller to use SSL and your own certificates, before installing OAA, OARM, and OUA. 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.nodePorts.http=30777 --set
controller.service.nodePorts.https=30443 --set
    controller.config.use-forwarded-headers=true --set
    controller.config.enable-underscores-in-headers=true --set
    controller.admissionWebhooks.enabled=false stable/ingress-nginx
```

> ✏️ **Note:**
>
> This will install the controller on https port 30443

C.2.2 Updating the Install Properties File for Installing OAA Using Ingress

To install OAA to use your HTTPS ingress controller, 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:30443
```

> **📝 Note:**
>
> - `install.global.serviceurl` should equal the value of the node and port that the ingress controller was installed on. If you are have something like a load balancer or OHS that will front end the ingress, then set it to the fully qualified hostname to the front-end.

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.

C.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
/oua-admin-ui
/oua/rui
/oua/drss
```

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

D

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

D.1 Viewing the Details of Database Tables

Learn about the specifics of each database table.

Topics

- VCRYPT_USERS
- 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
- VR_RULE_LOGS
- VCRYPT_ALERT

D.1.1 VCRYPT_USERS

Discover the specifics of the VCRYPT_USERS database table.

Description: Stores user information, including login ID, group ID, and creation/update timestamps. This table references the VCRYPT_USER_GROUPS table.

Database table name: VCRYPT_USERS

Primary Key: *USER_ID*

Database Column Name	Database Column Type	Description	Length
USER_ID (PK)	BIGINT	Auto-generated user ID.	16
EXT_USER_ID	VARCHAR	Auto-generated unique user identifier used to correlate users with an external user store.	255
LOGIN_ID		Refers to user login ID.	
GROUP_ID	BIGINT	Refers to the group to which the user belongs. It references VCRYPT_USER_GROUPS# GROUP_ID.	
CREATE_TIME	DATETIME	Timestamp when the user record was created.	6
UPDATE_TIME	TIMESTAMP	Timestamp when the user record was updated.	6

D.1.2 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	Auto generated group ID	16	-
GROUP_NAME	VARCHAR	Name of the group used in Create Context APIs	4000	-
DESCRIPTION	VARCHAR	Description for this group	4000	-
CREATE_TIME	DATETIME	Timestamp indicating when the user record was created	6	-
UPDATE_TIME	TIMESTAMP	Timestamp indicating when the user record was updated	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	VARCHAR	Note	4000	-

D.1.3 VCRYPT_TRACKER_USERNODE_LOGS

Discover the specifics of the VCRYPT_TRACKER_USERNODE_LOGS database table.

Description: Stores session information related to the user's activity, such as IP address, unique session ID (request_id), user login and group ID, actions performed, authentication activity score, and other session-related data. You can use this table to retrieve session and action information for a specific user based on a given time range.

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	Auto generated sequence ID	16

Database Column Name	Database Column Type	Description	Length
REQUEST_ID	VARCHAR	ID of the request. It is used to correlate user activity within the OARM system and is automatically generated each time a context is created. It is also referred to as contextID. It is referenced by VT_SESSION_ACTION _MAP and VR_RULE_LOGS.	256
EXT_SESSION_ID	VARCHAR	External session ID	512
CLIENT_DEVICE_ID	VARCHAR	ID of the device which is generated by the application.	256
REMOTE_IP_ADDR	BIGINT	Refers to client's IP address, which is stored in LONG format. For example, a client with IP address `123.221.111.101` is saved as `2078109541`.	15
BASE_IP_ADDR	BIGINT	Refers to the client's IP address, which is stored by dropping the last octet and converting it to LONG format. For example, a client with IP address `123.221.111.101` is saved as `2078109440`. It is referenced by VCRYPT_IP_LOCATION _MAP.	15
CLIENT_APPLICATION		Name of the client application provided when creating the context.	
NODE_ID	BIGINT	ID of the nodeID.	16
TRACKER_NODE_HIST ORY_ID	BIGINT	ID of the Tracker Node History (if available).	16
USER_ID	VARCHAR	Automatically generated unique user identifier or user ID for linking users with an external user store.	256
USER_LOGIN_ID	VARCHAR	Login ID of the user.	256
USER_GROUP_ID	VARCHAR	Group name of the user.	256
USER_SUB_GROUP_I D	VARCHAR	Sub GroupID of the user if available.	256

ORACLE

Database Column Name	Database Column Type	Description	Length
AUTH_STATUS	INT	Status of the authentication. The value corresponds to one of the options in `auth.status.enum`.	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
AUTH_CLIENT_TYPE_CODE	INT	Refers to the type of authentication method used. The value corresponds to one of the options in `auth.client.type.enum`.	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

Database Column Name	Database Column Type	Description	Length
PREAUTH_SCORE	INT	Pre Authentication score	8
POST_SCORE	INT	Score derived from the user authentication policy.	8
PREAUTH_ACTION	VARCHAR	Pre Authentication action	256
POST_ACTION	VARCHAR	Actions resulting from the user authentication policy.	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
CHALLENGE_CACHE	CLOB	Challenge cache data	-

D.1.4 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

ORACLE

Database Column Name	Database Column Type	Description	Length
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	VARCHAR	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	VARCHAR	Cache.	4000

D.1.5 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 node ID.	16
REQUEST_ID	VARCHAR	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

Database Column Name	Database Column Type	Description	Length
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	VARCHAR	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

D.1.6 VT_SESSION_ACTION_MAP

Discover the specifics of the VT_SESSION_ACTION_MAP database table.

Description: This table stores information about actions generated as part of the rules. Actions are based on the request_Id from (VCRYPT_TRACKER_USERNODE_LOGS) and user activity type. This table helps in identifying all actions and scores related to sessions and custom activities.

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	VARCHAR	Request ID or context ID. Referenced by VCRYPT_TRACKER_USERNODE_LOGS:REQUEST_ID.	256

Database Column Name	Database Column Type	Description	Length
TRX_ID	BIGINT	ID of the custom activity used for processing the rules. When it is not null, it refers to VT_TRX_LOGS#LOG_ID.	16
RUNTIME_TYPE	INT	User or custom activity type. The value is retrieved from `profile.type.enum`.	6
ACTION	VARCHAR	Action result for the user activity. The value is retrieved from `rule.action.enum`.	256
ORIGINAL_ACTION	VARCHAR	This was the original action, which got overridden finally.	256
OVERRIDE_REASON	INT	Override reason.	-
ACTION_LIST	VARCHAR	List of actions resulting from the execution of rules for the request or user activity.	256
SCORE	INT	Score result from user authentication policy.	-
IS_FINAL_ACTION	CHAR	Is this final action.	-
EXEC_TIME_MS		Time taken for rule execution, measured in milliseconds.	
RULE_TRACE_FP_ID		Fingerprint ID for the mapping of executed rules.	

D.1.7 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 User Group.	16	-
EXT_USERGROUP_ID	VARCHAR	External User group ID.	255	-
DESCRIPTION	VARCHAR	Description for this group.	2000	-
CREATE_TIME	DATETIME	Date/time creation of this user.	6	-

Database Column Name	Database Column Type	Description	Length	Enum values
UPDATE_TIME	TIMESTAMP	Date value.	6	-
USER_LIST_ID	BIGINT	ID of the user list.	16	-
USERGROUP_ST ATUS_CODE	INT	Status of the User group.	2	• STATUS_ACTI VE • STATUS_DISA BLED • STATUS_DEL ETED
NOTES	VARCHAR	Note.	4000	-

D.1.8 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	VARCHAR	Hash value for the fingerprint.	512
DATA_VALUE	VARCHAR	Data value for the fingerprint.	4000

D.1.9 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	VARCHAR	Value of the attribute.	256

ORACLE

D.1.10 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	VARCHAR	Value of the attribute.	256

D.1.11 VCRYPT_COUNTRY

Discover the specifics of the `VCRYPT_COUNTRY` database table.

Description: This table stores country-specific information provided by the geo data provider..

Database table name: `VCRYPT_COUNTRY`

Primary Key: *COUNTRY_ID*

Database Column Name	Database Column Type	Description	Length
COUNTRY_ID (PK)	BIGINT	An auto-generated unique identifier for the country.	16
COUNTRY_CODE	VARCHAR	Country code provided by the geo-location provider.	64
COUNTRY_NAME	VARCHAR	Country name provided by the geo-location provider.	4000
CREATE_TIME	TIMESTAMP	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	VARCHAR	Notes for this country.	4000

D.1.12 VCRYPT_STATE

Discover the specifics of the `VCRYPT_STATE` database table.

Description: This table stores information about states provided by the geo data provider.

Database table name: `VCRYPT_STATE`

Primary Key: *STATE_ID*

Database Column Name	Database Column Type	Description	Length
STATE_ID (PK)	BIGINT	An auto-generated unique identifier for the state. It references `VCRYPT_CITY#STATE_ID` and `VCRYPT_IP_LOCATION_MAP#STATE_ID`.	16
COUNTRY_ID	BIGINT	Refers to the country associated with this state. It references `VCRYPT_COUNTRY#COUNTRY_ID`.	16
STATE_CODE	VARCHAR	State code provided by the geo-location provider.	64
STATE_NAME	VARCHAR	Name of the state provided by the geo-location provider.	4000
CREATE_TIME	TIMESTAMP	Date/time for this log.	6
UPDATE_TIME	TIMESTAMP	Last update time for this object.	6
NOTES	VARCHAR	Notes for this state.	4000

D.1.13 VCRYPT_CITY

Discover the specifics of the `VCRYPT_CITY` database table.

Description: This table stores information about city provided by the geo data provider..

Database table name: `VCRYPT_CITY`

Primary Key: *CITY_ID*

Database Column Name	Database Column Type	Description	Length
CITY_ID (PK)	BIGINT	An auto-generated unique identifier for the city. It references `VCRYPT_IP_LOCATION_MAP#CITY_ID`.	16
STATE_ID	BIGINT	Refers to the state associated with this city. It references `VCRYPT_STATE#STATE_ID`.	16
CITY_CODE	VARCHAR	City code provided by the geo-location provider.	64
CITY_NAME	VARCHAR	Name of the city provided by the geo-location provider.	4000

Database Column Name	Database Column Type	Description	Length
CREATE_TIME	TIMESTAMP	Date/time for this log.	6
UPDATE_TIME	TIMESTAMP	Last update time for this object.	6
LATITUDE	VARCHAR	Latitude information provided by the geo-location provider.	20
LONGITUDE	VARCHAR	Longitude information provided by the geo-location provider.	20
TIMEZONE	VARCHAR	Time zone information provided by the geo-location provider.	20
NOTES	VARCHAR	Notes for this city.	4000

D.1.14 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	VARCHAR	Name of the ISP.	4000
CREATE_TIME	TIMESTAMP	Date/time for this log.	6
UPDATE_TIME	TIMESTAMP	Last update time for this object.	6

D.1.15 VCRYPT_IP_LOCATION_MAP

Discover the specifics of the VCRYPT_IP_LOCATION_MAP database table.

Description: This table stores data based on IP addresses and their associated geographic location details. It is imported from the location provider's geo-data.

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

Database Column Name	Database Column Type	Description	Length
FROM_IP_ADDR	BIGINT	Refers to the starting IP address for a specific geo-location, which is stored in LONG format. For example, the IP address 100.102.34.0 is saved as 1684414976.	15
TO_IP_ADDR	BIGINT	Refers to the ending IP address for a specific geo-location, which is stored in LONG format. For example, a client with IP address `100.102.34.0` is saved as `1684414976`.	15
CREATE_TIME	TIMESTAMP	Date/time for this log.	-
UPDATE_TIME	TIMESTAMP	Last update time for this object.	-
COUNTRY_ID	BIGINT	Refers to the country details associated with this geo-location record. It references `VCRYPT_COUNTRY#COUNTRY_ID`.	16
STATE_ID	BIGINT	Refers to the state details associated with this geo-location record. It references `VCRYPT_STATE#STATE_ID`.	16
CITY_ID	BIGINT	Refers to the city details associated with this geo-location record. It references `VCRYPT_CITY#CITY_ID`.	16
METRO_ID	BIGINT	ID of the metro for this IP.	16
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	Refers to the information about the connection type for this record. The possible values are derived from the `connection.type.enum` enum.	10

Database Column Name	Database Column Type	Description	Length
CONNECTION_SPEED	INT	Refers to the information about the connection speed for this record. The possible values are derived from the `location.linespeed.enum` enum.	10
TOP_LEVEL_DOMAIN	VARCHAR	Top level domain.	25
SEC_LEVEL_DOMAIN	VARCHAR	Second level domain.	128
ASN	VARCHAR	ASN	25
CARRIER	VARCHAR	Refers to the service provider details for the data.	128
ZIP_CODE	VARCHAR	Postal code for this location.	24
DMA	INT	U.S. Designated Market Area, AC Nielsen.	6
MSA	INT	Metropolitan Statistical Area.	6
PMSA	INT	Primary Metropolitan Statistical Area.	6
REGION_ID	BIGINT	ID of the region.	16
PHONE_AREA	VARCHAR	Phone area code.	10
IS_SPLIT	NUMBER	Is the IP split. If so, in some queries, we might have to do additional checks.	1
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

D.1.16 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	-

Database Column Name	Database Column Type	Description	Length	Enum Values
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	VARCHAR	Name for transaction.	4000	-
LABEL_RBKEY	VARCHAR	Resource bundle key for the name.	4000	-
DESCRIPTION	VARCHAR	Description of the object.	4000	-
DESC_RBKEY	VARCHAR	Resource bundle key for the description.	4000	-
TRX_DEF_KEY	VARCHAR	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	VARCHAR	Note for this object.	4000	-

D.1.17 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	-

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	VARCHAR	Name for transaction.	4000	-
LABEL_RBKEY	VARCHAR	Resource bundle key for the name.	4000	-
DESCRIPTION	VARCHAR	Description of the object.	4000	-
DESC_RBKEY	VARCHAR	Resource bundle key for the description.	4000	-
TRX_DEF_KEY	VARCHAR	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	VARCHAR	Note for this object.	4000	-

D.1.18 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	-

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	VARCHAR	Name for entity. For example address, customer.	4000	-
LABEL_RBKEY	VARCHAR	Resource bundle key for the name.	4000	-
DESCRIPTION	VARCHAR	Description of the object.	4000	-
DESC_RBKEY	VARCHAR	Resource bundle key for the description	4000	-
ENTITY_DEF_KEY	VARCHAR	Key of the entity. For example, 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	VARCHAR	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_PARA MS	VARCHAR	Static parameters to be passed to the Java class for name generation. For example, is a delimiter of ','	4000	-
NOTES	VARCHAR	Note for this object.	4000	-

D.1.19 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	VARCHAR	Name for the map.	4000	-
LABEL_RBKEY	VARCHAR	Resource bundle key for the name.	4000	-
DESCRIPTION	VARCHAR	Description of the object.	4000	-
DESC_RBKEY	VARCHAR	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	VARCHAR	Type of the relation.	4000	-
DISP_ORDER	INT	Display order.	6	-
STATUS	INT	Status	2	• STATUS_ACTI VE • STATUS_DISA BLED • STATUS_DEL ETED
NOTES	VARCHAR	Note for this object.	4000	-

D.1.20 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	VARCHAR	Name for the map.	4000
LABEL_RBKEY	VARCHAR	Resource bundle key for the name.	4000
DESCRIPTION	VARCHAR	Description of the object.	4000
DESC_RBKEY	VARCHAR	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	VARCHAR	Type of the relation.	4000
DISP_ORDER	INT	Display order.	6
NOTES	VARCHAR	Note for this object.	4000

D.1.21 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	-

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	VARCHAR	Name for data definition.	4000	-
LABEL_RBKEY	VARCHAR	Resource bundle key for the name.	4000	-
DESCRIPTION	VARCHAR	Description of the object.	4000	-
DESC_RBKEY	VARCHAR	Resource bundle key for the description.	4000	-
DATA_DEF_KEY	VARCHAR	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	VARCHAR	Note for this object.	4000	-

D.1.22 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	-

Database Column Name	Database Column Type	Description	Length	Enum Values
GLOBAL_ID	VARCHAR	Unique identifier which is used in import and export feature.	255	-
DEF_KEY	VARCHAR	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	VARCHAR	Name for column.	4000	-
LABEL_RBKEY	VARCHAR	Resource bundle key for the name.	4000	-
DESCRIPTION	VARCHAR	Description of the object.	4000	-
DESC_RBKEY	VARCHAR	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.	-	-
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.	-	-

Database Column Name	Database Column Type	Description	Length	Enum Values
DATA_TYPE	INT	Type of the data (numeric/ alphanumeric types).	-	-
DATA_FORMAT	VARCHAR	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	VARCHAR	Static parameters to be passed to the Java class for name generation.	4000	-
IS_AUTO_CREATED	CHAR	Whether this auto created.	-	-
NOTES	VARCHAR	Note for this object.	4000	-

D.1.23 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	VARCHAR	Name for the map.	4000
LABEL_RBKEY	VARCHAR	Resource bundle key for the name.	4000
DESCRIPTION	VARCHAR	Description of the object.	4000
DESC_RBKEY	VARCHAR	Resource bundle key for the description.	4000

Database Column Name	Database Column Type	Description	Length
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	VARCHAR	Type of the relation ("data", "name," and so on).	4000
NOTES	VARCHAR	Note for this object.	4000

D.1.24 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	-
GLOBAL_ID	VARCHAR	Unique identifier which is used in import and export feature.	255	-
LABEL	VARCHAR	Name for this data map.	4000	-
LABEL_RBKEY	VARCHAR	Resource bundle key for the name.	4000	-
DESCRIPTION	VARCHAR	Description of the object.	4000	-
DESC_RBKEY	VARCHAR	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.	-	-

Database Column Name	Database Column Type	Description	Length	Enum Values
TRANS_PARAMS	VARCHAR	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	-
RELATION_TYPE	VARCHAR	Type of the relation.	4000	-
STATUS	INT	Status	2	• STATUS_ACTIVE • STATUS_DISABLED • STATUS_DELETED
NOTES	VARCHAR	Note for this object.	4000	-

D.1.25 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	VARCHAR	Name for this data map.	4000	-
LABEL_RBKEY	VARCHAR	Resource bundle key for the name.	4000	-
DESCRIPTION	VARCHAR	Description of the object.	4000	-
DESC_RBKEY	VARCHAR	Resource bundle key for the description.	4000	-
TRANS_SCHEME	INT	Scheme for translation.	-	-
TRANS_PARAMS	VARCHAR	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	-
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	VARCHAR	Note for this object.	4000	-

D.1.26 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	VARCHAR	Name for this data map.	4000	-
LABEL_RBKEY	VARCHAR	Resource bundle key for the name.	4000	-
DESCRIPTION	VARCHAR	Description of the object.	4000	-
DESC_RBKEY	VARCHAR	Resource bundle key for the description.	4000	-
TRANS_SCHEME	INT	Scheme for translation.	-	-
TRANS_PARAMS	VARCHAR	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	VARCHAR	Note for this object	4000	-

D.1.27 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

Database Column Name	Database Column Type	Description	Length
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	VARCHAR	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	VARCHAR	Action	256
TRX_FLAG	INT	Flagging this transaction	3
POST_PROCESS_STATUS	INT	Status of the post processing	5
POST_PROCESS_RESULT	INT	Status of the post processing	5
TRX_DATA	VARCHAR	Transaction data as name value pair.	4000
DATA1	VARCHAR	Data one	256
DATA2	VARCHAR	Data two	256
DATA3	VARCHAR	Data three	256
DATA4	VARCHAR	Data four	256
DATA5	VARCHAR	Data five	256
DATA6	VARCHAR	Data six	256
DATA7	VARCHAR	Data seven	256
DATA8	VARCHAR	Data eight	256
DATA9	VARCHAR	Data nine	256
DATA10	VARCHAR	Data ten	256

D.1.28 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

Database Column Name	Database Column Type	Description	Length
CREATE_TIME	DATETIME	Date/time when this object was created	6
UPDATE_TIME	TIMESTAMP	Last update time for this object	6
DATA1	VARCHAR	Data one	4000
DATA2	VARCHAR	Data two	4000
DATA3	VARCHAR	Data three	4000
DATA4	VARCHAR	Data four	4000
DATA5	VARCHAR	Data five	4000
DATA6	VARCHAR	Data six	4000
DATA7	VARCHAR	Data seven	4000
DATA8	VARCHAR	Data eight	4000
DATA9	VARCHAR	Data nine	4000
DATA10	VARCHAR	Data ten	4000
NUM_DATA0	BIGINT	Numeric data 0	38
NUM_DATA1	BIGINT	Numeric data 1	38
NUM_DATA2	BIGINT	Numeric data 2	38

D.1.29 VR_RULE_LOGS

Discover the specifics of the VR_RULE_LOGS database table.

Description: Stores information about rule executions, including their actions, scores, and alert templates.

Database table name: VR_RULE_LOGS

Primary Key: *RULE_LOG_ID*

Database Column Name	Database Column Type	Description	Length
RULE_LOG_ID (PK)	BIGINT	Auto generated rule log ID.	16
CREATE_TIME		Time when the rule log was created.	
RULE_MAP_ID	BIGINT	Refers to the rule information. It references VCRYPT_PROFILE_RULE_MAP#PROFILE_RULE_MAP_ID.	16
SCORE		Score generated by the rule.	
ACTION_LIST		List of actions resulting from the execution of rules for the request or user activity.	
ALERT_TEMPL_ID_LIST		Alert templates for generating alerts.	

D.1.30 VCRYPT_ALERT

Discover the specifics of the `VCRYPT_ALERT` database table.

Description: Stores the generated alerts, including their contents, source, and context. Use the `REQUEST_ID` and `SESS_ACTION_MAP_ID` in this table to look up for alerts generated for a specific session/context or user activity.

Database table name: `VCRYPT_ALERT`

Primary Key: *ALERT_ID*

Database Column Name	Database Column Type	Description	Length
ALERT_ID (PK)	BIGINT	Auto generated alert ID.	16
SESS_ACTION_MAP_ID	BIGINT	Correlation with the session action map for user activity.	16
ALERT_TYPE	INT	Refers to the type of alert, with possible values from the `alert.type.enum` property.	10
CREATE_TIME	TIMESTAMP	Time when the alert was created.	6
REQUEST_ID	VARCHAR	Request ID for the session.	256
TRX_LOG_ID	BIGINT	ID of the custom activity used for processing the rules.	16
PROFILE_RULE_MAP_ID	BIGINT	ID of the rule that generated the alert. It references `VCRYPT_PROFILE_RULE_MAP#PROFILE_RULE_MAP_ID`.	16
RUNTIME_TYPE	INT	User or custom activity type, with possible values from the `profile.type.enum` property.	6
USER_ID	VARCHAR	Auto-generated unique user identifier used to correlate users with an external user store.	256
USER_LOGIN_ID	VARCHAR	Refers to the user login ID.	256
REMOTE_IP_ADDR	BIGINT	The IP address of the client, which is stored in LONG format. For example, the IP address 123.221.111.101 is saved as 2078109541.	15
ALERT_MESSAGE	VARCHAR	Text of the alert message.	4000

ORACLE

D.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))
```

ORACLE®

D.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

D.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.

D.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

D.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 D-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.

D.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;
```

D.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>
```

D.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;
```

D.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>
```

E
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

E.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

E.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.

E.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.

E.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.

E.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

E.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.

E.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, OARM, and OUA

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 Obtaining the Installation Software.

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.

E.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, OARM, and OUA

1. Download the OAA/OARM installation files to the new system. See Obtaining the Installation Software.

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.

E.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: Creating 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.

ORACLE®

5. Start the OAA Management Container by following: Creating the Management Container.

6. Run the OAA install script from inside the OAA Management container. See Deploying OAA, OARM, and OUA. This will create a new deployment based on your restored OAA/OARM file system data.

E.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>"
```

F

Configuring OMA Push Notifications Using Legacy FCM API's

The following configuration steps show how to configure push notifications for Android devices using Legacy FCM API's.

> ✎ **Note:**
>
> Administrators should be aware of the following:
>
> - Google is deprecating Legacy FCM API's in June 2024 and migrating to HTTP v1 API's.
>
> - For all new configurations it is recommended to use HTTP v1 API's. Steps to configure using HTTP v1 API's can be found in Configuring Oracle Mobile Authenticator Push Notification for Android.
>
> - If you have configured push notifications for Android in releases prior to OAA June 24 refresh, you will be using Legacy FCM API's. Administrators should migrate to HTTP v1 API's by upgrading to OAA June 24 refresh or later. The steps to upgrade and migrate to HTTP v1 API's can be found in Upgrading OAA, OARM, and OUA.
>
> - The steps in this sections below are for reference only.

Topics:

- Creating a Google Firebase Project Enabled for Google Cloud Messaging Using Legacy FCM APIs
- Configuring OAA Properties for Android Push Notification using Legacy FCM APIs

F.1 Creating a Google Firebase Project Enabled for Google Cloud Messaging Using Legacy FCM APIs

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 **Add project**.

3. In the **Project name** field, enter a name for your project. For example, `OAAAndroidPUSH`.

4. On the Google Analytics for your Firebase project page, deselect **Enable Google Analytics for this project**, and then click **Create project**.

5. Click **Continue** when your new project is ready.

6. In the left navigation pane of the project window, click the **Settings** icon, and then select **Project settings**.

7. On the **Project settings** page, click **Cloud Messaging**.

8. Under **Cloud Messaging API (Legacy)**, click on the ellipsis and select **Manage API in Google Cloud Console**.

9. In the new tab that appears, under **Cloud Messaging** click **ENABLE**.

10. Return to the original tab where you clicked **Cloud Messaging API (Legacy)** and refresh the page.

11. Note the values present in the **Server key** and **Sender ID** fields. These values are required later in Configuring OAA Properties for Android Push Notification using Legacy FCM APIs .

F.2 Configuring OAA Properties for Android Push Notification using Legacy FCM APIs

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 F-1   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.pushPreferencesEndpoint	The host and port used for push factor registration. This host and port should be accessible from the device. This corresponds to the host and port referenced in the SpuiUrl (`SpuiUrl=https://<host:port>/oaa/rui`) in Printing Deployment Details.	https://oaainstall
bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.challengeAnswerEndpoint	The host and port used for push factor runtime. This host and port should be accessible from the device. This corresponds to the host and port referenced in the Push URL (`Push=https://<host:port>/oaa-push-factor`) in Printing Deployment Details.	https://oaainstall

Table F-1   (Cont.) OAA Properties

Property Name	Description	Sample Value
bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.retrycount	Maximum number of unsuccessful retries of the challenge. Beyond this count the challenge is locked. The default value is 10. If you are using push notifications with Oracle Universal Authenticator you must set this value to 50.	50
bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.google.firebase.serverKey	The Firebase **Server Key**. See Creating a Google Firebase Project Enabled for Google Cloud Messaging Using Legacy FCM APIs.	AAAAh1hlXa8:APA91....
bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.google.firebase.senderId	The Firebase **Sender ID**. See Creating a Google Firebase Project Enabled for Google Cloud Messaging Using Legacy FCM APIs.	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.pushPreferencesEndpo
    int",
    "value": "https://oaainstall"
    },
    {
    "name":
    "bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.challengeAnswerEndpo
    int",
    "value": "https://oaainstall"
    },
    {
    "name": "bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.retrycount",
    "value": "50"
    },
    {
    "name":
    "bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.google.firebase.serv
    erKey",
    "value":
    "AAAAh1hlXa8:APA91bGOGR4pMYe9GC6a2rU169hTCBVmc................................
    LpU2F8_Egn7IZguC1Rr2HSNnROzXu1d1Lam0TJ"
    },
    {
    "name":
    "bharosa.uio.default.challenge.type.enum.ChallengeOMAPUSH.google.firebase.send
    erId",
    "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.

Next Steps: Registering the User Account with Oracle Mobile Authenticator for Android.