Oracle® Communications

Unified Inventory and Topology Deployment Guide

Release 8.0.1
G50242-01
April 2026

Oracle Communications Unified Inventory and Topology Deployment Guide, Release 8.0.1

G50242-01

Copyright © 2023, 2026, Oracle and/or its affiliates.

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

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

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

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

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Copyright © 2023, 2026, Oracle and/or its affiliates.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Copyright © 2023, 2026, Oracle and/or its affiliates.

8 Deploying the Active Topology Automation Service

Copyright © 2023, 2026, Oracle and/or its affiliates.

9 Upgrading ATA

Copyright © 2023, 2026, Oracle and/or its affiliates.

10 Localization

11 Deploying Service Impact Analysis

12 Dynamic Attribute Mapping between UIM and ATA

Copyright © 2023, 2026, Oracle and/or its affiliates.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Copyright © 2023, 2026, Oracle and/or its affiliates.

About This Content

This guide describes how to deploy and administer Oracle Communications Unified Inventory and Topology in a cloud native environment.

Audience

This document is for system administrators, database administrators, and developers who install and configure Unified Inventory and Topology.

Documentation Accessibility

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

Access to Oracle Support

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

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.

Copyright © 2023, 2026, Oracle and/or its affiliates.

1

About Unified Inventory and Topology

Unified Inventory and Topology includes the following services:

- Unified Inventory Management (UIM)
- Active Topology Automation (ATA)
- Authentication
- Authorization
- OpenSearch
- SmartSearch
- Unified Operations Message Bus
- Service Impact Analysis

ATA, Unified Operations Message Bus, Authentication, Authorization, Service Impact Analysis, and SmartSearch are cloud native containerized applications that are supported in a Kubernetes environment. UIM can be a traditional application or a cloud native instance.

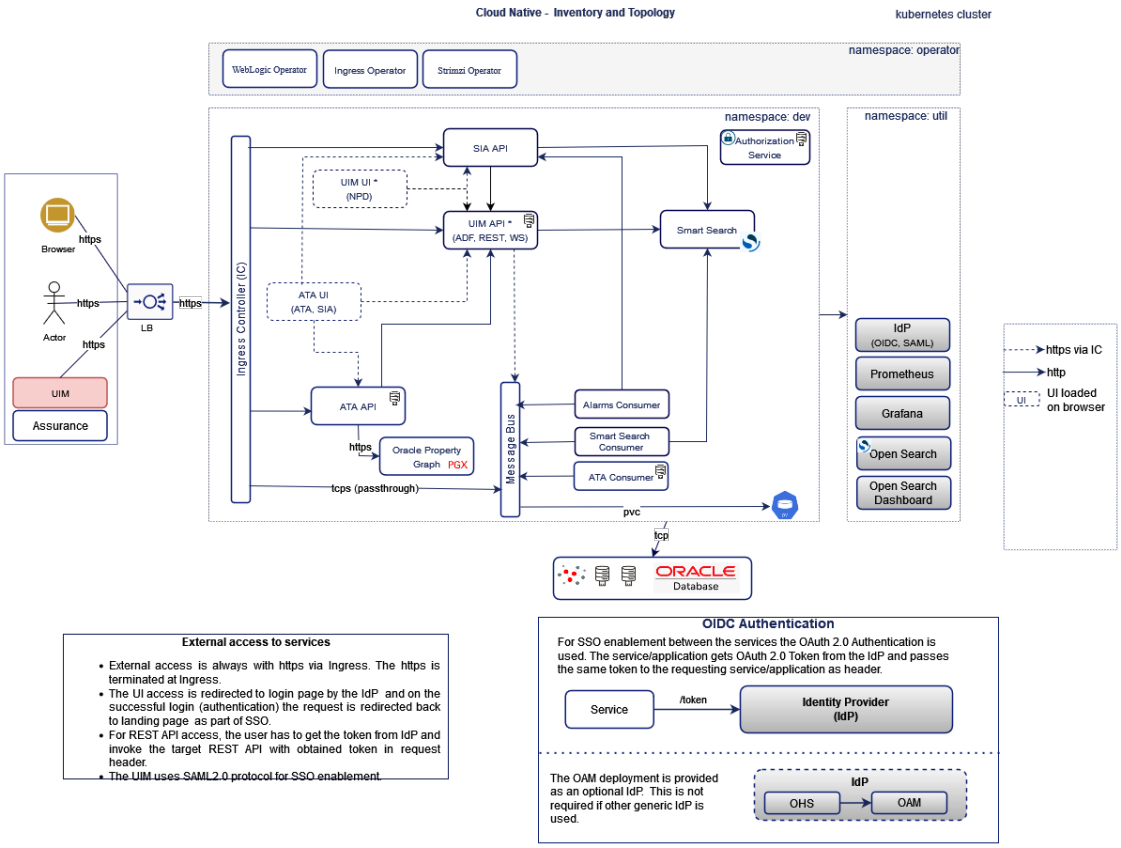
- The embedded topology from UIM is now available as a micro-service (ATA) based on Helidon MP.
- The communication between UIM and the ATA service is asynchronous and this is achieved by using Message Bus service.
- Deploying Service Impact Analysis is dependent on ATA deployment. You must deploy ATA and create an ATA instance before deploying Service Impact Analysis.

Unified Inventory and Topology Architecture

The following figure shows a high-level architecture of Unified Inventory and Topology and how the services communicate.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Figure 1-1 High-level Architecture of Unified Inventory and Topology



See the corresponding architecture diagrams of the services for more information.

About UIM

UIM is a standards-based telecommunications inventory management application that enables you to model and manage customers, services, and resources. UIM supports complex business relationships and provides full life-cycle management of services and resources. UIM provides you with a real-time, unified view of customers, services, and resource inventory, enabling you to develop and introduce new services quickly and cost-effectively. UIM supports two deployment models: traditional (on-premise) deployment and cloud native deployment in a Kubernetes cluster.

About ATA

Active Topology Automation (ATA) enables you to view the service, network, and resource topologies in the form of topology graphs. ATA uses Oracle Property Graph DB to manage the topology hierarchy.

ATA enables you to view the service, network, and resource topologies in the form of topology graphs. ATA uses Oracle Property Graph DB to manage the topology hierarchy.

ATA has the following sub components.

- ATA API

Copyright © 2023, 2026, Oracle and/or its affiliates.

- ATA PGX
- ATA Consumer
- ATA or Service Impact Analysis UI
- Alarm Consumer
- SmartSearch Consumer
- Service Impact Analysis
- Message Reconciliation

See *Active Topology Automation and Service Impact Analysis User's Guide* for more information.

About Serivce Impact Analysis

Service Impact Analysis enables you to view the Assurance events associated with Inventory resources and view the impacts to customer, service, network, logical and physical resources, and connectivity.

Service Impact Analysis also enables you to assign ownership to specific individuals and track the impact lifecycle through the analysis process.

About Authentication

Authentication leverages SAML 2.0 (Security Assertion Markup Language) and OpenId Connect (OIDC) authentication protocol of Identity Provider (IdP) to implement the Single Sign-On (SSO) authentication solution with the services (UIM, ATA, Service Impact Analysis, Authorization, Message Bus, SmartSearch, OpenSearch). This enables you to seamlessly access multiple applications without being prompted to authenticate for each application separately. The main advantage of SSO is that you are authenticated only once, which is when you log in to the first application and you do not need to authenticate again when you subsequently access different applications within the same web browser session.

IdP also supports the single logout (SLO) feature. If you access multiple applications using SSO within the same web browser session, and then if you log out of any one of the applications, you are logged out of all of the applications.

Examples for authentication services are: IDCS, Keycloak, and so on.

For more information about IDCS, see https://www.oracle.com/technical-resources/articles/middleware/oracle-identity-cloud-service.html

About Authorization

Authorization service defines a simplified and centralized approach for managing the authorization configurations for Unified Inventory and Topology services by defining the authorization policies. Authorization is the process of granting or denying access to specific resources based on the verified identity of a user whereas authentication is about verifying the identity of the user.

Authorization service is designed to provide permissions to access resources of an application for the authenticated user(s) with allowed role(s) or group(s).

Copyright © 2023, 2026, Oracle and/or its affiliates.

> **Note**
>
> Authorization service is not responsible to define users or assign role(s) or group(s) to the users.

Authorization service provides the capability to define various roles or groups in an application and define the permissions to resources under each role. For more information on Authorization, see "Deploying Authorization Service".

About OpenSearch

OpenSearch is a NoSQL database. It is an open-source search and analytics suite that makes it easy to ingest, search, visualize, and analyze data.

About SmartSearch

SmartSearch is a micronaut application, when integrated with OpenSearch, offers a powerful, flexible, and feature-rich search experience that can be tailored to specific business and user needs. Using OpenSearch as the underlying engine, SmartSearch can handle large volumes of data, perform real-time indexing, and support complex querying to enhance search relevancy. Features such as autocomplete, fuzzy matching, synonym recognition, and intelligent ranking make it easier for users to locate precise information, even if search terms are partially matched or misspelled.

About Unified Operations Message Bus

Message Bus is a distributed event store and stream-processing service. Message Bus service sends and receives events and messages asynchronously to a specific destination (called as **Topic**) between the services. The Message Bus service uses Apache Kafka, which is a distributed event store and stream-processing platform, as the messaging platform. For packaging or deploying, Strimzi is used. Strimzi simplifies the process of running Apache Kafka in a Kubernetes cluster. Strimzi also provides container images and operators for running Kafka on Kubernetes.

Planning UIM Installation

The following workflow helps you with information required for UIM installation.

Copyright © 2023, 2026, Oracle and/or its affiliates.

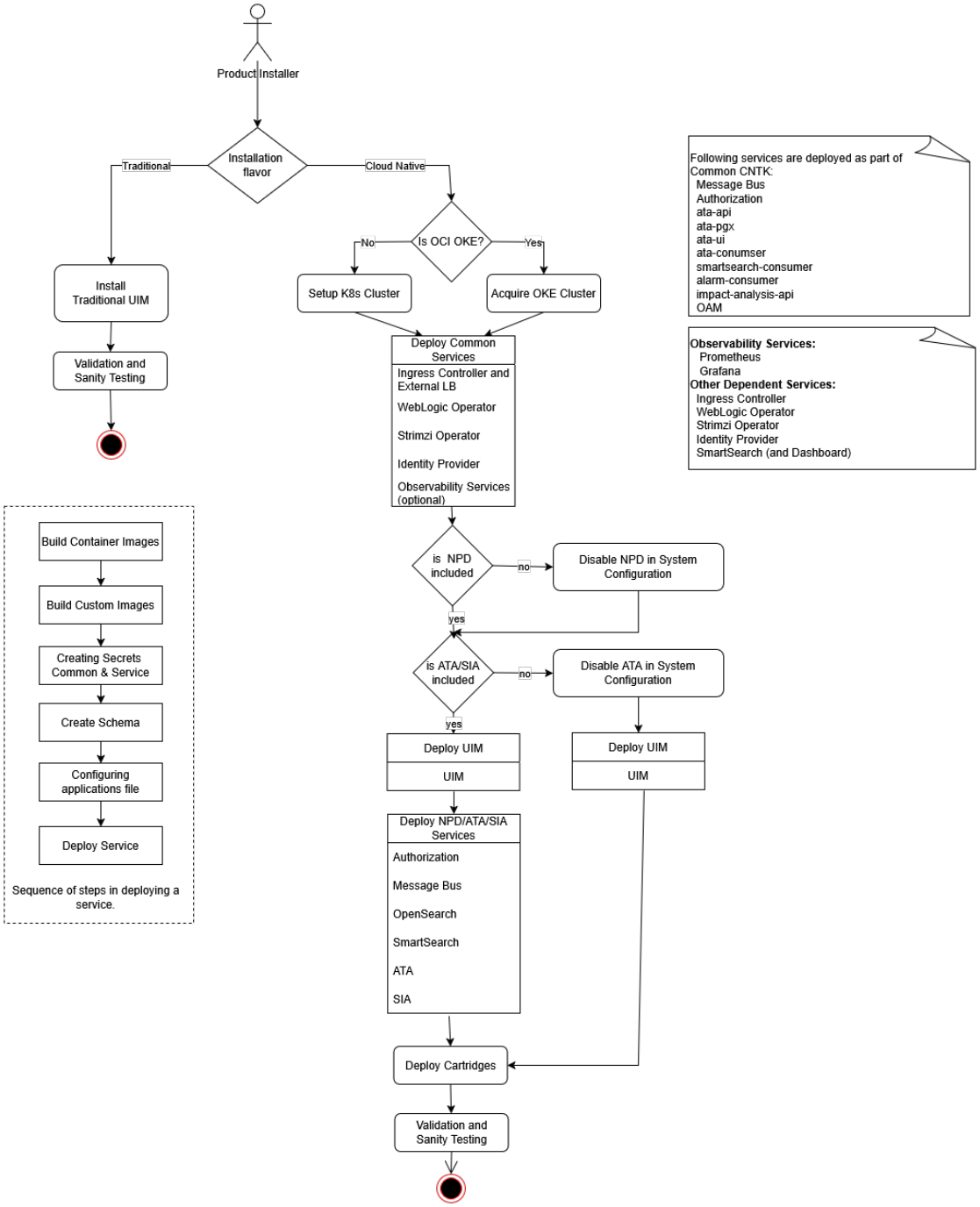


Table 1-1 Planning UIM Installation Workflow

Workflow Action	Reference	Description
Install Traditional UIM	Unified Inventory Management Installation Overview	Provides information on installing traditional UIM using an on-premise installer.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Table 1-1 (Cont.) Planning UIM Installation Workflow

Workflow Action	Reference	Description
Setup Kubernetes Cluster	Planning and Validating Your Cloud Environment Installing Oracle Property Graph Plugin in Database https://strimzi.io/docs/operators/latest/deploying#considerations-for-data-storage-str https://kubernetes.io/docs/concepts/storage/storage-classes/	To deploy cloud native services, you must set up and validate a list of prerequisite software. Before starting the service deployments: • Install property graph plugins on the PDB that are used for ATA. • Configure the Storage Class in Kubernetes to provision Persistent Volumes dynamically to be used for the Message Bus service and OpenSearch service.
Deploy Common Services	About the Unified Inventory and Topology Toolkit About the UIM Cloud Native Toolkit	Download the required software and set the environment variables.
Ingress Controller and External Load Balancer	About Load Balancing and Ingress Controller Installing the Ingress Controller SSL Certificates	You can use any Ingress Controller that conforms to the standard Kubernetes ingress API and that supports annotations required for UIM. Samples for HAProxy are included in the toolkit. For a secure access of services, you must set up an Ingress Controller with TLS termination. TLS Termination setup for services is provided in deployment of each service.
WebLogic Operator	Installing the WebLogic Kubernetes Operator Container Image	The WebLogic Kubernetes Operator (**operator**) supports running your WebLogic Server and Fusion Middleware Infrastructure domains on Kubernetes.
Strimzi Operator	Strimzi Operator	Required only if ATA or NPD should be enabled. The Strimzi Operator supports deployment of Apache Kafka cluster on Kubernetes or OpenShift.
Identity Provider	Configuring Authentication for Services	You can use any Identity Provider (IdP) that supports SAML 2.0 (Security Assertion Markup Language) and OIDC (OpenId Connect) authentication protocols for implementing SSO (Single Sign-On) authentication solution among services.
Observability Services	Setting Up Prometheus and Grafana Deploying OpenSearch and OpenSearch Dashboard	Optionally, deploy services such as Grafana, Prometheus and OpenSearch for Metrics and Log Monitoring.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Table 1-1 (Cont.) Planning UIM Installation Workflow

Workflow Action	Reference	Description

Disable NPD in System Configuration	Not applicable.	Modify **$SPEC_PATH/$PROJECT/$INSTANCE/config/uim/system-config/custom-config.properties** to disable NPD:
		
		`#If true, render new canvas in network visualization tab.`
		`uim.ui.createNetworkGuidedFlow.canvas.enabled=false`
		`openSearchEnabled=false`

Disable ATA in System Configuration	Not applicable.	Modify **$SPEC_PATH/$PROJECT/$INSTANCE/config/uim/system-config/custom-config.properties** to disable ATA:
		
		`# Topology MicroService`
		`disableTopology=true`
		`microServiceEnabled=false`

Deploy UIM		
Build Container Images	Creating UIM Cloud Native Images	
Build Custom Images	Customizing Images	This is optional. Required while extending the base image.
Create Secrets	Creating Secrets	
Create DB Schema	Installing the UIM and RCU Schemas	
Update Application Configurations	Setting System Properties	
Deploy Service	Creating a Basic UIM Instance	
Deploy NPD/ATA Services		
Authorization	Deploying Authorization Service	
Message Bus	Deploying Unified Operations Message Bus	
OpenSearch	Deploying OpenSearch and OpenSearch Dashboard	Oracle OCI OpenSearch has to be used in OKE Cluster environment.
SmartSearch	Deploying SmartSearch	

ATA	Deploying the Active Topology Automation Service	If NPD is not required, update **$SPEC_PATH/<PROJECT>/<INSTANCE>/shapes/<shapeName>/ata.yaml** to set SmartSearch Consumer `replicaCount` to zero.
		
		`smartSearchConsumer:`
		`  name: "smartsearch-consumer"`
		`  replicaCount: 0`

Service Impact Analysis	Deploying Service Impact Analysis	
Validation and Sanity Testing	Validation and Sanity Testing	

Copyright © 2023, 2026, Oracle and/or its affiliates.

Validation and Sanity Testing

To perform validation and sanity testing:

1. Log in to UIM using `https://<instance>.<project>.<hostSuffix>:<LB Port>/Inventory`.
2. If NPD is enabled, select **Create Network**. The **Verify Guided Flow** appears
   a. Enter **Network Name**.
   b. Select **Create and Save New Location**.
   c. Associate **New Location** to **Network** and click **Continue**.
   d. Add **New Resource** - **Logical Device** to **Location**.
   e. Select **Continue** and then click **Finish**. The **Verify Network** page appears.
3. If ATA is enabled, open ATA using `https://<instance>.<project>.topology.<hostSuffix>:<LB Port>/apps/ata-ui`
   a. Select **Create New** - **Device**.
   b. Enter new device name (from above) and click **Search**.
   c. Verify the device that appears.

Planning UIM Upgrade

This section provides information about planning your UIM upgrade for traditional and cloud native environments.

Planning Traditional UIM Upgrade

The following workflow helps you with information required for UIM upgrade.

Copyright © 2023, 2026, Oracle and/or its affiliates.

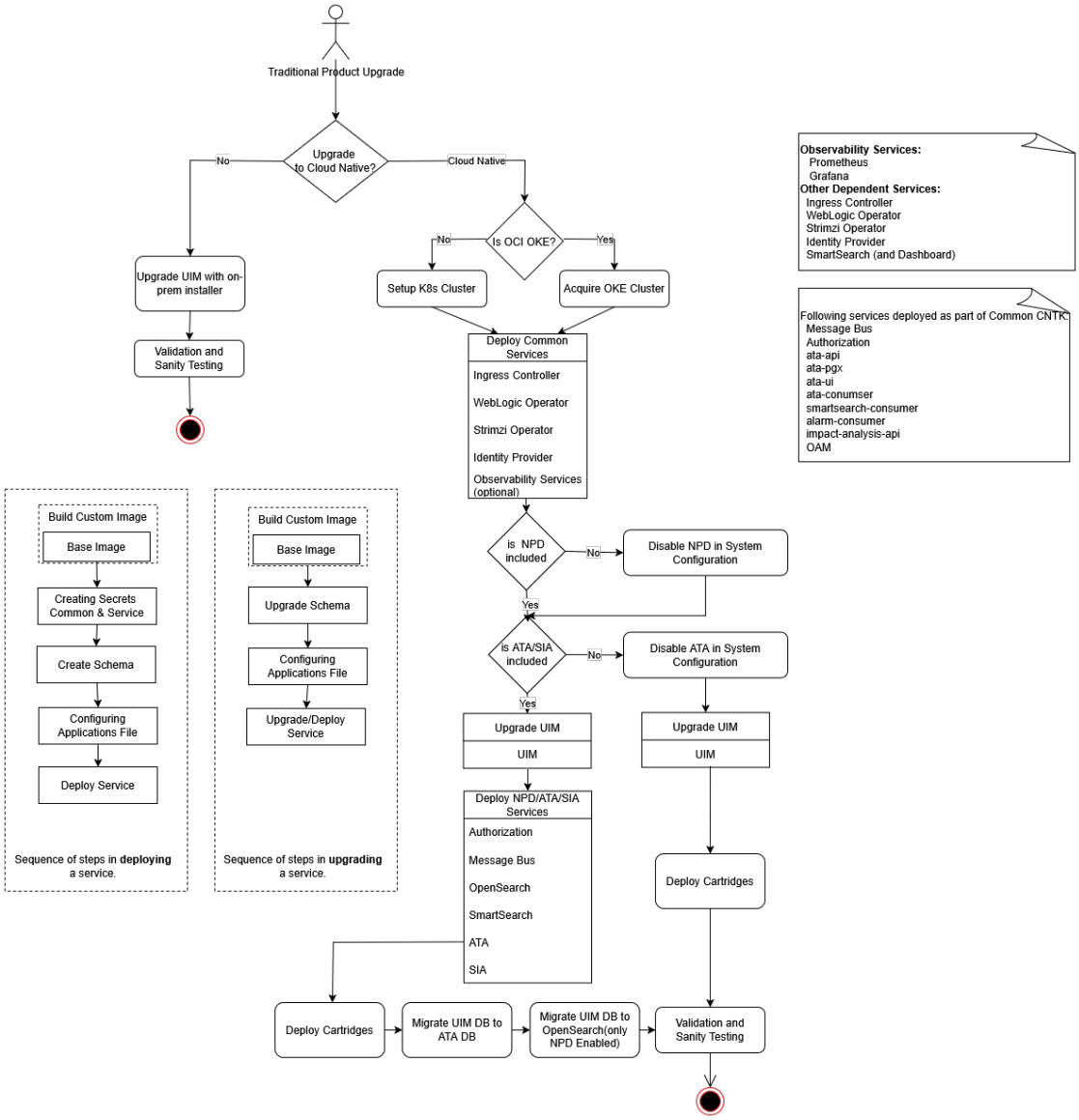


Table 1-2 Planning Traditional UIM Upgrade Workflow

Workflow Action	Reference	Description
Upgrade Traditional UIM	Upgrading Unified Inventory Management	Provides information on installing traditional UIM using an on-premise installer.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Table 1-2 (Cont.) Planning Traditional UIM Upgrade Workflow

Workflow Action	Reference	Description
Setup K8s Cluster	Planning and Validating Your Cloud Environment Installing Oracle Property Graph Plugin in Database https://strimzi.io/docs/operators/latest/deploying#considerations-for-data-storage-str	To deploy cloud native services, you must set up and validate a list of prerequisite software. Before starting the service deployments: • Install property graph plugins on the PDB that are used for ATA. • Configure the Storage Class in Kubernetes to provision Persistent Volumes dynamically to be used for the Message Bus service and OpenSearch service.
Deploy Common Services	About the Unified Inventory and Topology Toolkit About the UIM Cloud Native Toolkit	Download the required software and set the environment variables.
Ingress Controller and External Load Balancer	About Load Balancing and Ingress Controller Installing the Ingress Controller SSL Certificates	You can use any Ingress Controller that conforms to the standard Kubernetes ingress API and that supports annotations required for UIM. Samples for HAProxy are included in the toolkit. For a secure access of services, you must set up an Ingress Controller with TLS termination. TLS Termination setup for services is provided in deployment of each service.
WebLogic Operator	Installing the WebLogic Kubernetes Operator Container Image	The WebLogic Kubernetes Operator (**operator**) supports running your WebLogic Server and Fusion Middleware Infrastructure domains on Kubernetes.
Strimzi Operator	Strimzi Operator	Required only if ATA or NPD should be enabled. The Strimzi Operator supports deployment of Apache Kafka cluster on Kubernetes or OpenShift.
Identity Provider	Configuring Authentication for Services	Not applicable.
Observability Services	Setting Up Prometheus and Grafana Deploying OpenSearch and OpenSearch Dashboard	Optionally, deploy services such as Grafana, Prometheus and OpenSearch for Metrics and Log Monitoring.

Disable NPD in System Configuration	Not applicable.	Modify **$SPEC_PATH/$PROJECT/$INSTANCE/config/uim/system-config/custom-config.properties** to disable NPD:
		`#If true, render new canvas in network visualization tab.`
		`uim.ui.createNetworkGuidedFlow.canvas.enabled=false`
		`openSearchEnabled=false`

Copyright © 2023, 2026, Oracle and/or its affiliates.

Table 1-2 (Cont.) Planning Traditional UIM Upgrade Workflow

Workflow Action	Reference	Description
Disable ATA in System Configuration	Not applicable.	Modify **$SPEC_PATH/$PROJECT/$INSTANCE/config/uim/system-config/custom-config.properties** to disable ATA: `# Topology MicroService disableTopology=true microServiceEnabled=false`
Upgrade UIM	Moving to UIM Cloud Native from a Traditional Deployment Creating the UIM Cloud Native Images Creating a Basic UIM Cloud Native Instance Upgrading ATA	Not applicable.
Deploy UIM		Not applicable.
Build Container Images	Creating UIM Cloud Native Images	Not applicable.
Build Custom Images	Customizing Images	This is optional. Required while extending the base image.
Create Secrets	Creating Secrets	Not applicable.
Create DB Schema	Installing the UIM and RCU Schemas	Not applicable.
Update Application Configurations	Setting System Properties	Not applicable.
Deploy Service	Creating a Basic UIM Instance	Not applicable.
Deploy NPD/ATA Services		Not applicable.
Authorization	Deploying Authorization Service	Not applicable.
Message Bus	Deploying Unified Operations Message Bus	Not applicable.
OpenSearch	Deploying OpenSearch and OpenSearch Dashboard	Oracle OCI OpenSearch has to be used in OKE Cluster environment.
SmartSearch	Deploying SmartSearch	Not applicable.

ATA	Deploying the Active Topology Automation Service	If NPD is not required, update **$SPEC_PATH/<PROJECT>/<INSTANCE>/shapes/<shapeName>/ata.yaml** to set SmartSearch Consumer `replicaCount` to zero.
		
		`smartSearchConsumer:`
		`  name: "smartsearch-consumer"`
		`  replicaCount: 0`

Service Impact Analysis	Deploying Service Impact Analysis	Not applicable.
UIM	Overview of the UIM Cloud Native Deployment	After UIM CN instance is created, redeploy the **inventory.ear** from admin console to update the new roles for NPD.
Migrate UIM DB to ATA DB	Installing ATA Service Schema Dynamic Data Mapping from UIM	ATA DB Schema has to be created before migrating UIM Data to Graph DB.

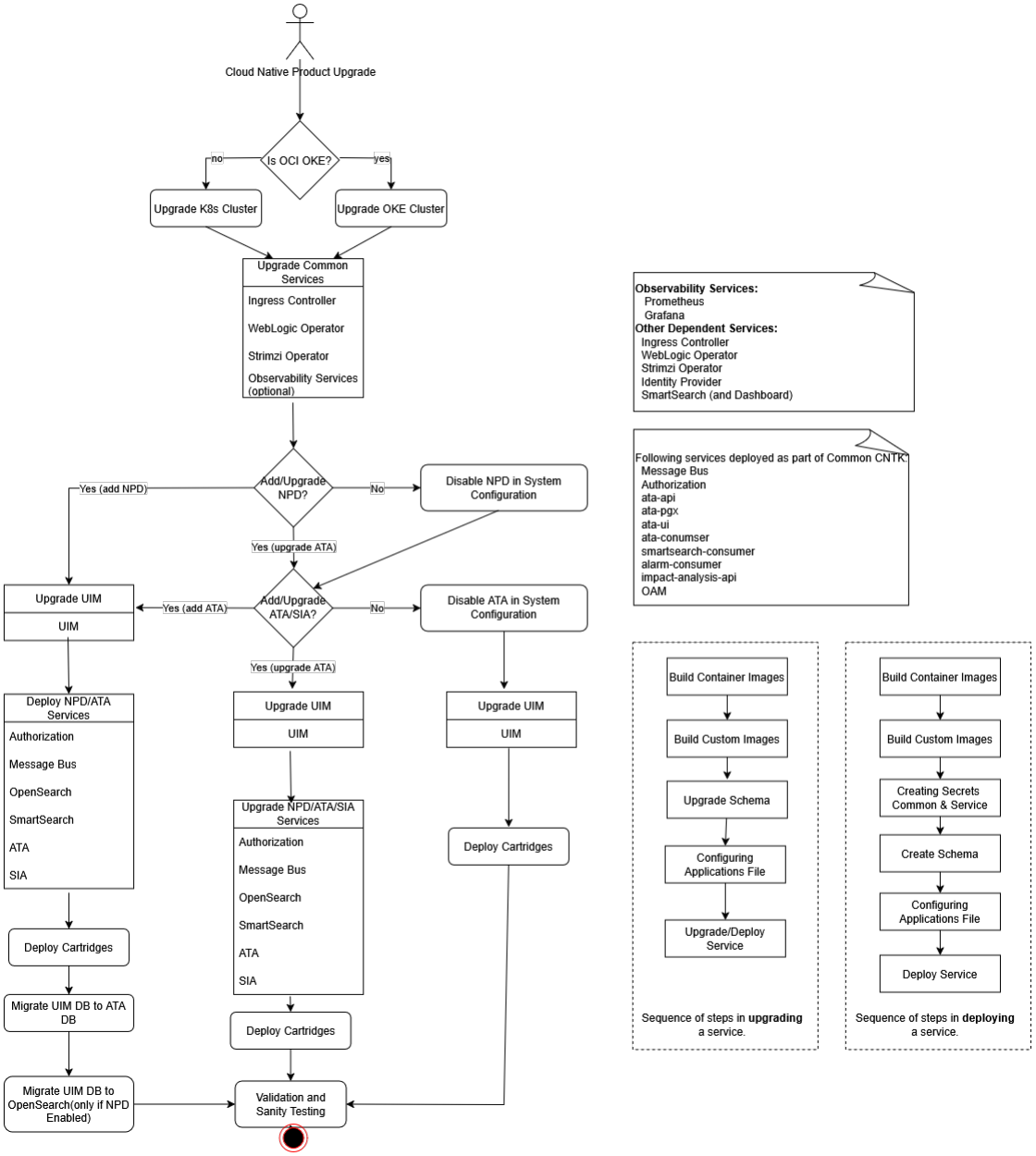
Copyright © 2023, 2026, Oracle and/or its affiliates.

Table 1-2 (Cont.) Planning Traditional UIM Upgrade Workflow

Workflow Action	Reference	Description
Migrate UIM DB to OpenSearch	Data Migration and Dynamic Attribute Mapping between UIM and SmartSearch	Data from UIM DB to OpenSearch NoSQL DB will be migrated with the help of SmartSearch and OpenSearch services.
Validation and Sanity Testing	Validation and Sanity Testing	Not applicable.

Planning UIM Cloud Native Upgrade

The following workflow helps you with information required for UIM Cloud Native upgrade.



Copyright © 2023, 2026, Oracle and/or its affiliates.

Table 1-3 Planning UIM Cloud Native Upgrade Workflow

Workflow Action	Reference	Description
Upgrade Traditional UIM	Upgrading Unified Inventory Management	Provides information on installing traditional UIM using an on-premise installer.
Setup K8s Cluster	Planning and Validating Your Cloud Environment Installing Oracle Property Graph Plugin in Database https://strimzi.io/docs/operators/latest/deploying#considerations-for-data-storage-str	To deploy cloud native services, you must set up and validate a list of prerequisite software. Before starting the service deployments: • Install property graph plugins on the PDB that are used for ATA. • Configure the Storage Class in Kubernetes to provision Persistent Volumes dynamically to be used for the Message Bus service and OpenSearch service.
Deploy Common Services	About the Unified Inventory and Topology Toolkit About the UIM Cloud Native Toolkit	Download the required software and set the environment variables.
Ingress Controller and External Load Balancer	About Load Balancing and Ingress Controller Installing the Ingress Controller SSL Certificates	You can use any Ingress Controller that conforms to the standard Kubernetes ingress API and that supports annotations required for UIM. Samples for HAProxy are included in the toolkit. For a secure access of services, you must set up an Ingress Controller with TLS termination. TLS Termination setup for services is provided in deployment of each service.
WebLogic Operator	Installing the WebLogic Kubernetes Operator Container Image	The WebLogic Kubernetes Operator (**operator**) supports running your WebLogic Server and Fusion Middleware Infrastructure domains on Kubernetes.
Strimzi Operator	Strimzi Operator	Required only if ATA or NPD should be enabled. The Strimzi Operator supports deployment of Apache Kafka cluster on Kubernetes or OpenShift.
Identity Provider	Configuring Authentication for Services	Not applicable.
Observability Services	Setting Up Prometheus and Grafana Deploying OpenSearch and OpenSearch Dashboard	Optionally, deploy services such as Grafana, Prometheus and OpenSearch for Metrics and Log Monitoring.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Table 1-3 (Cont.) Planning UIM Cloud Native Upgrade Workflow

Workflow Action	Reference	Description

Disable NPD in System Configuration	Not applicable.	Modify **$SPEC_PATH/$PROJECT/$INSTANCE/config/uim/system-config/custom-config.properties** to disable NPD:
		
		`#If true, render new canvas in network visualization tab.`
		`uim.ui.createNetworkGuidedFlow.canvas.enabled=false`
		`openSearchEnabled=false`

Disable ATA in System Configuration	Not applicable.	Modify **$SPEC_PATH/$PROJECT/$INSTANCE/config/uim/system-config/custom-config.properties** to disable ATA:
		
		`# Topology MicroService`
		`disableTopology=true`
		`microServiceEnabled=false`

Upgrade UIM	Moving to UIM Cloud Native from a Traditional Deployment Creating the UIM Cloud Native Images Creating a Basic UIM Cloud Native Instance Upgrading ATA	Not applicable.
Deploy UIM		Not applicable.
Build Container Images	Creating UIM Cloud Native Images	Not applicable.
Build Custom Images	Customizing Images	This is optional. Required while extending the base image.
Create Secrets	Creating Secrets	Not applicable.
Create DB Schema	Installing the UIM and RCU Schemas	Not applicable.
Update Application Configurations	Setting System Properties	Not applicable.
Deploy Service	Creating a Basic UIM Instance	Not applicable.
Deploy NPD/ATA Services		Not applicable.
Authorization	Deploying Authorization Service	Not applicable.
Message Bus	Deploying Unified Operations Message Bus	Not applicable.
OpenSearch	Deploying OpenSearch and OpenSearch Dashboard	Oracle OCI OpenSearch has to be used in OKE Cluster environment.
SmartSearch	Deploying SmartSearch	Not applicable.
ATA	Deploying the Active Topology Automation Service	Not applicable.
Service Impact Analysis	Deploying Service Impact Analysis	Not applicable.
UIM	Overview of the UIM Cloud Native Deployment	After UIM CN instance is created, redeploy the **inventory.ear** from admin console to update the new roles for NPD.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Table 1-3 (Cont.) Planning UIM Cloud Native Upgrade Workflow

Workflow Action	Reference	Description
Migrate UIM DB to ATA DB	Installing ATA Service Schema Dynamic Data Mapping from UIM	ATA DB Schema has to be created before migrating UIM Data to Graph DB.
Migrate UIM DB to OpenSearch	Data Migration and Dynamic Attribute Mapping between UIM and SmartSearch	Data from UIM DB to OpenSearch NoSQL DB will be migrated with the help of SmartSearch and OpenSearch services.
Validation and Sanity Testing	Validation and Sanity Testing	Not applicable.

Installing Oracle Property Graph Plugin in Database

ATA uses Oracle Property Graph of Oracle Database that offers a powerful graph support to explore and discover complex relationships within ATA graphs.

Graph Server and Client is a software package that is required for Property Graph.

Creating Property Graph Roles

To create Property Graph roles, see Graph Developer's Guide for Property Graph https://docs.oracle.com/en/database/oracle/property-graph/24.4/spgdg/user-authentication-and-authorization.html#GUID-C006C651-DCA5-419D-859C-173840321408. Perform the steps 5 and 6 as a DBA on the database server to create the roles required by the graph server.

Copyright © 2023, 2026, Oracle and/or its affiliates.

2

About the Unified Inventory and Topology Toolkit

This chapter describes the components required for Unified Inventory and Topology.

Unified Inventory and Topology Toolkit

From Oracle Software Delivery Cloud, download the following:

- Oracle Communications Unified Inventory Management Common Toolkit
- Oracle Communications Unified Inventory Management Cloud Native Image Builder
- Oracle Communications Unified Inventory Management ATA Image Builder
- Oracle Communications Unified Inventory Management SmartSearch Image
- Oracle Communications Unified Inventory Management Authorization Image Builder

Perform the following tasks:

1. Copy the above downloaded archives into directory **workspace** and unzip the archives.
2. Export the unzipped path to the WORKSPACEDIR environment variable.
3. On Oracle Linux, where Kubernetes is hosted, download and extract the tar archive on each host. This host has a connectivity to the Kubernetes cluster.
4. Alternatively, on OKE, for an environment where Kubernetes is running, extract the contents of the tar archive (on each OKE client host). The OKE client host is the bastion host that is set up to communicate with the OKE cluster.

```
$ mkdir workspace
$ export WORKSPACEDIR=$(pwd)/workspace
//Untar UIM Builder
$ tar -xf $WORKSPACEDIR/uim-image-builder.tar.gz --directory $WORKSPACEDIR
//Untar ATA Builder
$ tar -xf $WORKSPACEDIR/ata-builder.tar.gz --directory $WORKSPACEDIR
//Untar Authorization Builder
$tar -xf $WORKSPACEDIR/authorization-builder.tar.gz --
directory $WORKSPACEDIR
//Untar Common Toolkit
$ tar -xf $WORKSPACEDIR/common-cntk.tar.gz --directory $WORKSPACEDIR
$ export COMMON_CNTK=$WORKSPACEDIR/common-cntk
```

Assembling the Specifications

To assemble the specifications:

1. Create a directory (either in local machine or version control system where the deployment pipelines are available) to maintain the specification files needed to deploy the service. Export the directory to SPEC_PATH environment variable.

Copyright © 2023, 2026, Oracle and/or its affiliates.

2. Export the environment variables as follows:

```
export COMMON_CNTK=$WORKSPACEDIR/common-cntk
export SPEC_PATH=<path to directory created in step 1>
export STRIMZI_NS=<namespace for strimzi deployment> ex. strimzi
export PROJECT=<k8s namespace where you planned to deploy the services>
ex. sr
export INSTANCE=<the instance name to be used for your services> ex. quick
```

3. Run the following command to assemble the required specification files:

```
$COMMON_CNTK/scripts/assemble-specifications.sh -p $PROJECT -i $INSTANCE -
s $SPEC_PATH
```

   If the script runs successfully without any errors, verify the configuration files by checking the contents of the **$SPEC_PATH** directory.

> **Note**
>
> All scripts in the Common cloud native toolkit are designed to read the required configuration files only from the path specified by **$SPEC_PATH**.

4. Copy other specification files as required:
   - Persistent volumes and persistent volume claims files from $COMMON_CNTK/samples/nfs
   - Role and role bindings from $COMMON_CNTK/samples/rbac
   - Credential files from $COMMON_CNTK/samples/credentials

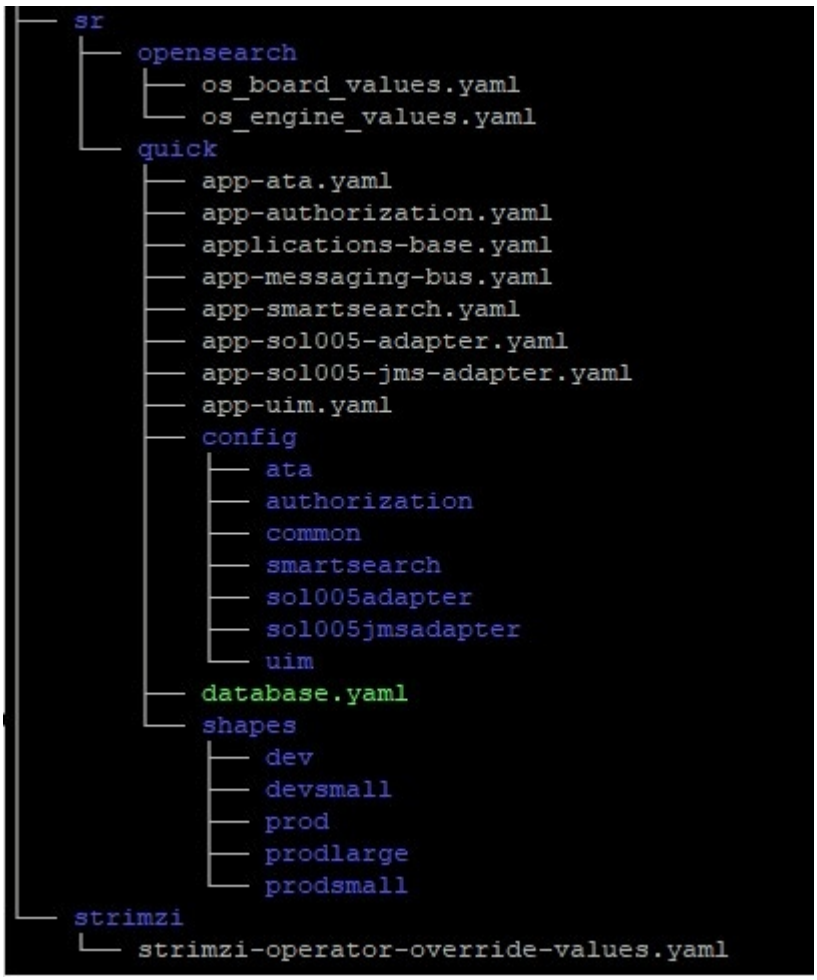
About the Specifications File

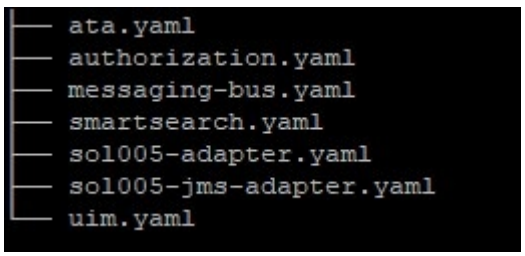
After assembling the specifications, it generates a directory structure with the following files:

- **$SPEC_PATH/$PROJECT/$INSTANCE**: This file contains all specifications files (**applications-base.yaml** and **app-<serviceName>.yaml**) for the services along with the **shapes** and **config** directories.
- **$SPEC_PATH/$PROJECT/$INSTANCE/shapes**: This file contains the resource configuration files for each service. The immediate subdirectories represent shape names, and each of these contains **<serviceName>.yaml** files that define the hardware resource specifications. New directories can also be added here, by following the same pattern, with shape-specific YAML files for all services. By default, the shape configurations are for **devsmall**, **dev**, **prodsmall**, **prod**, and **prodlarge** shapes.
- **$SPEC_PATH/$PROJECT/$INSTANCE/config**: This file contains applications and logging configuration files for all the services, if you want to change or provide any application-level configuration you need to edit in this directory.
- **$SPEC_PATH/$PROJECT/$INSTANCE/config/common**: This file contains the common configuration files that are common across all services.
- **$SPEC_PATH/$PROJECT/opensearch**: This file contains the files required for OpenSearch deployment configuration.
- **$SPEC_PATH/$STRIMZI_NS**: This file contains the files required for Strimzi operator deployment to override the default configuration.

Copyright © 2023, 2026, Oracle and/or its affiliates.

> **Note**
>
> All scripts in the Common cloud native toolkit are designed to read the required configuration files only from the path specified by $SPEC_PATH.





Details within the Specification Files

You can find the following details in the corresponding specification files:

Copyright © 2023, 2026, Oracle and/or its affiliates.

- **applications-base.yaml**: This file contains the common deployment configuration for all the services like selection and configuration of ingressController, loadbalancer port, storageVolume, gcLogs and so on. The contents of this file are applicable to all services.
- **app-<serviceName>.yaml**: This file contains the specific deployment configurations for each service such as image details , affinity configurations, and so on. The contents of this file are only applicable to the corresponding service.
- **<shapeName>/<serviceName>.yaml**: This file contains the hardware resources' configurations such as CPU, memory, replica counts, heap memory configuration, and so on.
- **database.yaml**: This file contains the configurations such as database image, storageVolume, tablespace details, and so on, for performing any operations on database schemas of the services.

Customizing the Shapes

The predefined shapes: **devsmall, dev, prodsmall, prod,** and **prodlarge** are available at **$SPEC_PATH/$INSTANCE/$PROJECT/shapes**. Each directory contains the <serviceName>.yaml files that define the hardware resources for that service.

The **devsmall** shape includes the smallest size defined and it does not include the requests and limits defined. Therefore, pods do not fail to schedule if there are a limited number of CPUs or memory, but as this shape does not have any upper limit set, the shape can grow to a larger size at the runtime.

Use the predefined shapes as much as possible. As per your requirement, you can create your own shape and use it with all services.

To create the new shape (for example: **customShape**):

1. Create a new directory **customShape** at **$SPEC_PATH/$PROJECT/$INSTANCE/shapes** as follows:

   ```
   cd $SPEC_PATH/$PROJECT/$INSTANCE/shapes/
   mkdir customShape
   ```

2. Copy the files from the predefined shape **dev** as follows:

   ```
   cp dev/*.yaml customShape/
   ```

3. Edit and change the configurations of <serviceName>.yaml files in the **customShape** directory as per your requirement.
4. Edit **$SPEC_PATH/$PROJECT/$INSTANCE/applications-base.yaml** and provide **customShape** as the shape value:

   ```
   shape: customShape
   ```

   When you create or upgrade any service as per the values defined in **customShape** files, the corresponding service will be configured.

Image Builders

The following image builders are required to build the corresponding services for an end-to-end integrated environment:

Copyright © 2023, 2026, Oracle and/or its affiliates.

- UIM Image Builder: This includes archive `uim-image-builder.tar.gz`, which is required to build UIM, UIM DB Installer Images. See "Creating the UIM Cloud Native Images" in *UIM Cloud Native Deployment Guide* for more information.
- Authorization Builder: This includes `authorization-builder.tar.gz`, required to build Authorization images. For more information, see "Creating Authorization Images".
- ATA Builder: This includes `ata-builder.tar.gz`, required to build ATA API, ATA UI, ATA PGX, ATA Consumer, and the ATA DB Installer images. See "Prerequisites for Creating ATA Images" for more information.

All builder toolkits include manifest files and scripts to build the images.

About the Manifest File

A manifest file can be found in directory path $WORKSPACEDIR/<*service-builder*>/bin/<*service*>_manifest.yaml. The manifest file describes the input that goes into the *service* images. It is consumed by the image build process. The default configuration in the latest manifest file provides all necessary components for creating the *service* images easily. *A service* can be ATA, Authorization, SmartSearch, OpenSearch, or UIM.

You can also customize the manifest file. This enables you to:

- Specify any Linux image as the base, as long as it is a binary and is compatible with Oracle Linux.
- Upgrade the Oracle Enterprise Linux version to a newer version to uptake a quarterly CPU.
- Upgrade the JDK version to a newer JDK version to uptake a quarterly CPU.
- Choose a different **userid** and **groupid** for **oracle:oracle user:group** that the image specifies. The default is 1000:1000.

> **Note**
>
> The `schemaVersion` and `date` parameters are maintained by Oracle. Do not modify these parameters. Version numbers provided here are only examples. The manifest file specifies the actual versions that Oracle recommends.

There are various sections in the manifest file such as:

- **Service Base Image**: The Service Base image is a necessary building block of the final *service* container images. However, it is not required by the *service* to create or manage any service instances.
  Linux parameter: The `Linux` parameter specifies the base Linux image to be used as the base Docker or Podman image. The version is the two-digit version from **/etc/redhat-release**:

  ```
  linux:
      vendor: Oracle
      version: 9-slim
      image: <container>/os/oraclelinux:9-slim
  ```

  The vendor and the version details are used for validating while an image is being built and for querying at run-time.

Copyright © 2023, 2026, Oracle and/or its affiliates.

> **Note**
>
> To troubleshoot issues, Oracle support requires you to provide these details in the manifest file used to build the image.

- The `userGroup` parameter that specifies the default **userId** and **groupId**:

```
userGroup:
  username: <username>
  userid: <userID>
  groupname: <groupname>
  groupid: <groupID>
```

- The `jdk` parameter that specifies the JDK vendor, version, and the staging path:

```
jdk:
    vendor: Oracle
    version: <jdk_version>
    path: $CN_BUILDER_STAGING/downloads/java/jdk-<jdk_version>_linux-
x64_bin.tar.gz
```

- The Tomcat parameter specifies the Tomcat version and its staging path.

> **Note**
>
> This is applicable only for the ATA service.

```
tomcat:
  version: <tomcat_version>
  path: $CN_BUILDER_STAGING/downloads/tomcat/apache-tomcat-
<tomcat_version>.tar.gz
```

- A serviceImage parameter, where **tag** is the tag name of the *service* image.

```
serviceImage:
  tag: latest
```

> **Note**
>
> See "UIM Software Compatibility" in *UIM Compatibility Matrix* for software versions.

Deployment Toolkits

The Common Cloud Native toolkit is required to deploy the services for an end-to-end integrated environment. It includes the **common-cntk.tar.gz** file that is required to deploy Authorization, ATA, SmartSearch, OpenSearch, UIM, and Message Bus services in the cloud native environment.

See "Creating a Basic UIM Cloud Native Instance" in *UIM Cloud Native Deployment Guide*, for more information.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Common Cloud Native Toolkit

The Common cloud native toolkit (Common CNTK) includes:

- Helm charts to manage the ATA, Authorization, SmartSearch, OpenSearch, UIM, and Message Bus services.
- Scripts to manage secrets for the services.
- Scripts to manage schemas for the services.
- Scripts to create, update, and delete the ATA, Authorization, SmartSearch, OpenSearch, UIM, and Message Bus services.
- Sample pv and pvc yaml files to create persistent volumes.
- Sample charts to install HAProxy.
- Scripts to register and un-register the namespaces with Strimzi and WebLogic operator.
- The **applications-base.yaml** file that contains the common configuration for all services like ingress controller, tls, authentication and storageVolume, and so on.
- The application-specific files **app-uim.yaml**, **app-ata.yaml**, **app-messaging-bus.yaml**, **app-authorization.yaml**, and **app-smartsearch.yaml** that contain the corresponding configuration options for individual services, such as image details, java options, and support overriding the common configurations defined in **applications-base.yaml**.
- The **database.yaml** file that contains the configurations that is used while creating, deleting, or upgrading schemas for services, such as database installer images, tablespace info, and so on.
- The hardware resource configuration files (**shape** directory**)** for the development and production environments.
- The **config** directory that contains the logging and application configurations for the services.
- The **strimzi-operator-override-values.yaml** file that enables you to override the configuration for deploying strimzi operator which is used for message bus service.

The **applications-base.yaml** and **database.yaml** files have common values that are applicable for all services in Common CNTK.

The values applicable to individual services are present in app-<service-name>.yaml files along with the values that are applicable for specific services.

For customized configurations to override the default values, update the values under the specific files at **$SPEC_PATH/$PROJECT/$INSTANCE**.

While running the scripts, the *project* and *instance* values should be provided, where *project* indicates the namespace of the Kubernetes environment where the service is deployed and *instance* is the identifier of the corresponding *service* instance, if multiples instances are created within the same namespace.

> **Note**
>
> As multiple instances of Message Bus cannot exist in the same namespace, only one instance is created for all services within the same namespace.

Copyright © 2023, 2026, Oracle and/or its affiliates.

While creating a basic instance for all these services, the project name is considered as **sr** and the instance name is considered as **quick**.

> **Note**
>
> - The project and instance names must not contain any special characters.
> - There are common values specified in the **applications-base.yaml** and **database.yaml** files for the services. To override the common value user can specify that value in the specific file of a service. If the value under in the specific file is empty, then common value is considered.

Deploying the Services

You must deploy and configure all services in the following sequence:

1. (Optional) Select your Identity provider for Authentication.

   > **Note**
   >
   > You can choose any Generic Identity Provider that support SAML 2.0 and OAUTH 2.0 protocols for authentication. The samples of Oracle IDCS Identity Provider are packaged with ATA.

2. (Optional) Create OAUTH 2.0 client which will be configured with the below services,
3. Deploy Authorization service.
4. Deploy Message Bus.
5. Deploy OpenSearch.
6. Deploy SmartSearch.
7. Deploy UIM (traditional or cloud native).
8. Configure Traditional UIM with Message Bus and ATA, and restart UIM. See "Setting System Properties" in *UIM System Administrator's Guide*, for more information.
9. Deploy ATA.

> **Note**
>
> Ensure that each individual service is deployed successfully and verified in the above mentioned order as there are dependencies between these services. Ensure that for production instance, for High Availability, the Message Bus is set up with at least 3 replicas for kafka-cluster.

Setting Up Prometheus and Grafana

Install the Prometheus Operator and Grafana using the **kube-prometheus-stack** helm charts available at Prometheus Community GitHub: https://github.com/prometheus-community/helm-charts.

Copyright © 2023, 2026, Oracle and/or its affiliates.

The `prometheus-community/kube-prometheus-stack` is a Helm chart maintained by the Prometheus Community. The `kube-prometheus-stack` gives you a production-ready Kubernetes monitoring stack with minimal effort, instead of installing Prometheus, Grafana, and Alertmanager separately and integrating them manually.

The following are sample commands provided for your reference. For more details, see the community GitHub on installing the `kube-prometheus-stack` chart.

Install kube-prometheus-stack

```
#Add the Helm repo
helm repo add prometheus-community https://prometheus-community.github.io/
helm-charts
helm repo update

#Install the chart with kube-prometheus-stack release name
helm install kube-prometheus-stack prometheus-community/kube-prometheus-stack
-n <prometheus-operator-namespace>

#Verify the pods
kubectl get pods -n <prometheus-operator-namespace>
```

Configuring Metrics for Services

The UIM and associated services (such as Kafka Message Bus, ATA, and SIA) are validated with the Prometheus Operator **PodMonitor** and **ServiceMonitor** custom resources, which enable the automatic configuration of metrics scrape jobs in Prometheus. After the Prometheus operator is successfully deployed, configure the metrics scrape jobs. The common CNTK includes sample Helm charts that automate this configuration using **PodMonitor** and **ServiceMonitor** resources.

The Helm chart is available at **$COMMON_CNTK/samples/charts/prometheus-monitor-resources**. See the accompanying **README.md** file for instructions on automating the configuration of metrics scrape jobs.

Overview:

1. Verify that all prerequisites are met.
2. Update the **values.yaml** file to match your target environment.
3. UIM metrics URL access needs credentials in Kubernetes secret. Validate the creation.
4. Install, upgrade, or uninstall the Helm release to automate the configuration of scrape jobs.

Install prometheus-monitor-resource chart

```
#Install the chart for configuring scrape jobs.
helm install <application-namespace>-<application-instance>-prometheus-
monitor-resource \
  $COMMON_CNTK/samples/charts/prometheus-monitor-resources \
  -n <prometheus-operator-namespace>

#verify the deployment of PodMonitors
kubectl get podmonitors -n <prometheus-operator-namespace>

#verify the deployment of ServiceMonitor
kubectl get servicemonitors -n <prometheus-operator-namespace>
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

Copyright © 2023, 2026, Oracle and/or its affiliates.

3
Configuring Authentication for Services

This chapter describes how to configure authentication for Unified Inventory and Topology services such as UIM, ATA, Message Bus, SmartSearch, Service Impact Analysis, Message Reconciliation, and OpenSearch.

About Authentication

This section provides instructions for setting up Single Sign-On (SSO) authentication for Unified Inventory and Topology services.

These services implement the Single Sign-On (SSO) authentication solution using OIDC protocol from any supported Identity Provider (IdP), which enables you to seamlessly access multiple applications without being prompted to authenticate for each application separately. The main advantage of SSO is that you are authenticated only once, which is when you log in to the first application and then you are not required to authenticate again when you subsequently access different applications. However, these applications must be with the same (or lower) authentication level (as the first application) and opened within the same web browser session.

These services also support the Single Logout (SLO) feature. If you access multiple applications using SSO within the same web browser session, and then if you log out of any one of the applications, you are logged out of all the applications.

> **Note**
>
> UIM requires an IdP that supports OIDC.

For more information about how to enable authentication, see *UIM Cloud Native Deployment Guide*. OIDC is supported for all other services such as Message Bus, ATA, Authorization, SmartSearch and OpenSearch.

Ensure IdP users are assigned below groups for ATA or Service Impact Analysis access:

- Groups for UIM:
  - `uim-users`: To access UIM, users must be assigned a role that belongs to the **uim-users** group. If a user does not have a role in the **uim-users** group, the following error appears after successful authentication:

    ```
    You do not have permission to access this page. Contact the
    Administrator.
    ```

    ```
    uimuser: UIM defines the uimuser role (application role in em console).
    This is a super role that grants access to all UIM resources, so the
    role should not be granted to everyone. Rather, Oracle recommends that
    you define your own application roles to restrict access to UIM
    resources. The uimuser role is part of the uim-users WebLogic Server
    group.
    ```


Copyright © 2023, 2026, Oracle and/or its affiliates.

- `SecureDataAccessGroup`: To control accessing encrypted data in UIM as follows:
  - Users assigned to the `SecureDataAccessGroup` role are authorized to decrypt and view encrypted data in UIM.
  - UIM Administrator users who are members of the `SecureDataAccessGroup` can enter and manage the encryption key within UIM.

  > **Note**
  >
  > You must have the correct encryption key to recover plain text from encrypted data. Back up your encryption key before using the service. If the encryption key is lost, you will be permanently unable to access encrypted data.

- `uimuser`: The `uimuser` role is an application role defined in the Enterprise Manager (EM) Console. It is a privileged role that provides access to all UIM resources. Therefore, this role should not be assigned to all users. Oracle recommends creating custom application roles to control and restrict access to UIM resources based on user responsibilities. The `uimuser` role is associated with the `uim-users` WebLogic Server group.
  For any new user, ensure the following:
  - The user must be assigned the `uim-users` WebLogic Server group to access the UIM user interface (UI).
  - The user must be assigned an appropriate application role in the EM Console to access UIM resources.

  For more information, see "Alternative Approach: Group-Based Access Management".

- Groups for ATA:
  - `AtaAdministrator`: This group has the privileges of **Administrator** and **AdvancedUser** roles. The users with this group can:
    - Create, view, and search topology graphs
    - Edit and delete all saved searches
    - Navigate to all summary pages
    - Create, view, edit, and delete icons and colors customization
  - `AtaAdvancedUser`: This group has the privileges of **AdvancedUser** role. The users with this group can:
    - Create, view, and search topology graphs
    - Edit and delete saved searches created by the corresponding user
    - Navigate to all summary pages
- Groups for Service Impact Analysis:
  - `SiaAdministrator` : This group has the privileges of **Administrator**, **SIA User**, and **SIA Advanced User** roles. The users with this group can edit, reject, assign, and delete all events or impact reports.
  - `SiaAdvancedUser`: This group has the privileges of **SIA Advanced User** role. The users with this group can edit, reject, assign, and delete events or impact reports for which the current user is the owner.
  - `SiaUser`: The users with this role can:
    - View, search, or filter events

Copyright © 2023, 2026, Oracle and/or its affiliates.

    - View impact summary, initiate analysis, and view entity in ATA or UIM
    - View and export impact reports
    - View resource summary
- Groups for Message Reconciliation:
  - `MessageReconciliationAdministrator`: This group has the privileges of **Administrator** and **Viewer** roles. The users with this group can edit, delete, rebuild, resubmit, and view all events.
  - `MessageReconciliationUser`: This group has the privileges of **Viewer** role. The users with this group can:
    - View or search events.
    - View events summary for consumers grouped by error code, actions, or state.

> **Note**
>
> The Message Reconciliation is a User Interface which is supported by the Fallout Events Resolution RESTful endpoints. These groups are applicable for both User Interface and RESTful endpoints.

To configure authentication for all other services, perform the steps mentioned in the following sections:

- Adding Common OAuth Secret and ConfigMap
- Common Configuration Options For all Services

Sample references are provided as Appendix, see the following content for the corresponding authentication type:

- Configuring KeyCloak as Identity Provider for UIM, ATA, and Message Bus
- Setting Up Unified Inventory Management for Single Sign-On Authentication

Alternative Approach: Group-Based Access Management

Instead of assigning roles directly to individual users, you can use a group-based approach:

1. Create a group in the WebLogic Embedded LDAP.
2. Assign the required application roles to this group in the EM Console.
3. Assign the `uim-users` group to the newly created group to enable UI access.
4. In your Identity Provider (IdP), create users and assign this group to them.

> **Note**
>
> Ensure that user and group names in EM Console and WebLogic match those in Keycloak. Keycloak enforces the following naming conventions:
>
> - Usernames are stored in lowercase (uppercase input is automatically converted).
> - Role names are stored with the first letter in uppercase, even if entered in lowercase.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Adding Common OAuth Secret and ConfigMap

To add `COMMON OAUTH` secret and `ConfigMap`, run the following script to create the OAuth configuration as secrets and ConfigMap:

```
$COMMON_CNTK/scripts/manage-app-credentials.sh -p sr -i quick -s $SPEC_PATH
create oauthConfig
```

Enter the values as prompted:

```
Provide Oauth credentials  for   'sr-quick'   ...
Client Id: topologyClient #Provide Client ID
Client Secret: xxxxx #Provide Client Secret
Identity Provider Uri: <Identity-Provider-Uri>
Client Scope: <oauth-client-scope>  (if scope is not configured for oidc-
client keep blank)
Client Audience: <oauth-client-audience> (if audience not configured for oidc-
client keep blank)
Token Endpoint Uri: <token_endpoint_uri> #Provide oauth token endpoint URI
Valid Issue Uri: <valid issue uri> #Provide the valid issue URI
Introspection Endpoint Uri: <introspection_endpoint_uri> #Provide the
Introspection Endpoint URI
JWKS Endpoint Uri: <JWKS_endpoint_uri> #Provide JWKS Endpoint URI
Cookie Name: <Cookie-Name>
Cookie Encryption Password: <Cookie-Encryption-Password>

Provide Truststore details ...
Certificate File Path (ex. ./idpcert.pem): ./idpcert.pem    #provide
Certificate file path
```

Sample for IDCS is as follows:

```
Provide Oauth credentials for 'sr-quick' ...
Client Id: xxxxxxxxxxxxx
Client Secret: xxxx-xxxx-xxxx-xxxx
Identity Provider Uri: https://<IDCS URL>:443
Client Scope: https://quick.sr.topology.uim.org:30443/first_scope
Client Audience: https://quick.sr.topology.uim.org:30443/
Token Endpoint Uri: https://<IDCS URL>:443/oauth2/v1/token
Valid Issue Uri: https://identity.oraclecloud.com/
Introspection Endpoint Uri: https://<IDCS URL>:443/oauth2/v1/introspect
JWKS Endpoint Uri: https://<IDCS URL>:443/admin/v1/SigningCert/jwk
Cookie Name: OIDC_SESSION
Cookie Encryption Password: <Cookie Encryption Password>

Provide Truststore details ...
Certificate File Path (ex. idpcert.pem): ./idpcert.pem #provide identity
provider certificate to be used by Message Bus
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

> **Note**
>
> The **oauthConfig** secret is used by Message Bus, ATA, SmartSearch, and Authorzation services. If you are creating them in different namespaces or instances, you need to create this secret in both namespaces or instances.

Common TrustStore Secret

To add Common TrustStore secret:

1. Run the following command to create or update truststore by passing the Identity Provider SSL certificate:

   ```
   keytool -importcert -v -alias <param> -file <path to IDP cert file> -
   keystore <truststorename>.jks -storepass <password>
   ```

   A sample is as follows:

   ```
   keytool -importcert -v -alias idpcert -file identityprovidercert.pem -
   keystore truststore.jks -storepass ****
   ```

   > **Note**
   >
   > You must add the corresponding certificates for UIM and Identity Providers. If the Identity Provider and UIM certificates are not common, add both in the same truststore.

2. Run the following command to generate **commonTrust** Kubernetes secret and provide the details appropriately when the system prompts for the truststore file path and password of truststore:

   ```
   $COMMON_CNTK/scripts/manage-app-credentials.sh -p $PROJECT -i $INSTANCE -
   s $SPEC_PATH create commonTrust

   $PROJECT -> This value should match with the namespace in which you want
   to run the helm install.
   $INSTANCE -> This value should be matching with the instance that you are
   going to specify while doing the helm install.

   Enter the values as prompted:
   Truststore Path: truststore.jks
   Truststore Passphrase: <passphrase>
   ```

3. Verify the following:

   ```
   $kubectl get secret -n sr
   sr-quick-oauth-credentials

   $kubectl get cm -n sr
   sr-quick-oauth-config-cm
   ```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
$kubectl get cm -n sr sr-quick-common-truststore
```

> **Note**
>
> The **commonTrust** secret is used by ATA, SmartSearch, and Authorization services. If you are creating them in different namespaces or instances, you should create this secret in all namespaces or instances.

Common Configuration Options For all Services

You can provide configurations that are common across all services in the **$SPEC_PATH/<PROJECT>/<INSTANCE>/config/common/common-config.yaml** file and run the **commonConfig** command.

You can use this option to provide any configuration for ATA (api, ui, impact-analysis-api) and Authorization service. The Mandatory Identity Provider configuration details are passed using **oauthConfig** secret. If you want to override that configuration or to supply any additional configuration, you can use this option.

> **Note**
>
> Before running the command, make sure you assemble the specifications. For more information, see "Assembling the Specifications".

```
#In case of IDCS as IdP, you have to provide the following additional
provider details
security:
  providers:
  - abac:
  - oidc:
  - idcs-role-mapper:
      multitenant: false #update this as per the IDCS instance used.
      oidc-config:
        client-id: "${security.properties.idp-client-id}"
        client-secret: "${security.properties.idp-client-secret}"
        identity-uri: "${security.properties.idp-uri}"
        audience: "${security.properties.idp-audience}"
```

Add a block for api configuration of users, which is similar to the existing **idcs-role-mapper** config block. Along with the **idcs-role-mapper** config, add the folowing configuration to **common-config.yaml** file:

```
security:
  properties:
      idp-query-users-scope: "urn:opc:idm:__myscopes__"
      idp-query-users-endpoint: "<Identity provider URI>/admin/v1/Users"
      idp-user-attributes:
         displayName: "Resources[].displayName"
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
userName: "Resources[].userName"
email: "Resources[].emails[0].value"
id: "Resources[].id"
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

4

Deploying Authorization Service

This chapter describes how to deploy and manage the Authorization service.

Overview

The Authorization service defines a simplified and centralized approach for managing the authorization configurations for services.

The consuming service (for example, ATA) sends an HTTP request with access token (from IdP) as Authorization Header to get the allowed resources for the authenticated users. Based on the response, users are allowed to perform certain actions on the consuming service.

For example, users with `AtaAdministrator` role are allowed to customize the icons and colors on the ATA canvas.

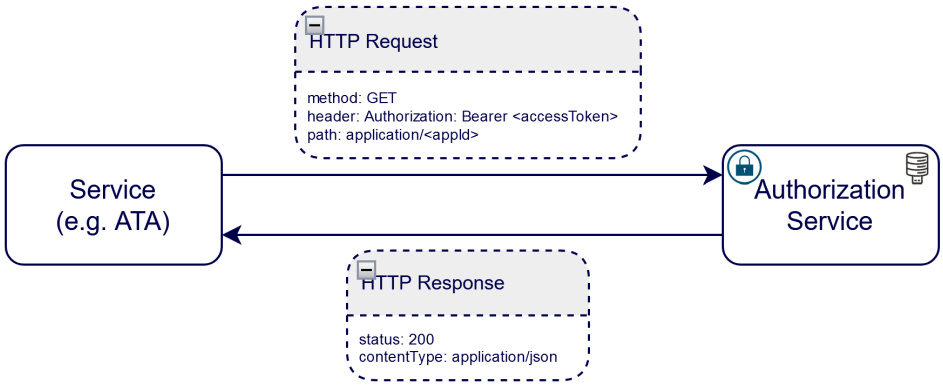
The users with `AtaAdvancedUser` role are allowed to access the advance settings of ATA canvas, but are not allowed to customize the icons and colors on the ATA canvas.

A new security role, `SecureDataAccessGroup`, controls access to encrypted data in UIM as follows:

- Users assigned to the `SecureDataAccessGroup` role are authorized to decrypt and view encrypted data in UIM.
- UIM Administrator users who are members of the `SecureDataAccessGroup` can enter and manage the encryption key within UIM.

> **Note**
>
> You must have the correct encryption key to recover plain text from encrypted data. Back up your encryption key before using the service. If the encryption key is lost, you will be permanently unable to access encrypted data.

The architecture diagram of Authorization service is as follows:



Copyright © 2023, 2026, Oracle and/or its affiliates.

Creating Authorization Images

You must install the prerequisite software and tools for creating the Authorization images.

Prerequisites for Creating Authorization Images

You require the following prerequisites for creating the Authorization images:

- Podman on the build machine (if Linux version is greater than or equal to 8).
- Docker on the build machine (if Linux version is lesser than 8).
- Authorization Builder Toolkit. For the toolkit, see "Unified Inventory and Topology Toolkit".
- Install Maven and update the path variable with **Maven Home** as follows:

  ```
  set PATH variable export PATH=$PATH:$MAVEN_HOME/bin
  ```

- Install Java with JAVA_HOME set in the environment as follows:

  ```
  Set PATH variable export PATH=$PATH:$JAVA_HOME/bin
  ```

- Bash, to enable the `<tab>` command.

> **Note**
>
> See UIM Software Compatibility for software versions.

Configuring Authorization Images

The corresponding manifest file describes the input that goes into the Authorization images. The default configuration in the latest manifest file provides the necessary components for creating the Authorization images. See "About the Manifest File" for more information.

Creating Authorization and Schema Images

To create the Authorization and schema images:

1. Set the **JAVA_HOME** variable in your environment to match the location of your Java installation as follows:

   ```
   export JAVA_HOME=<location of jdk-21>
   ```

2. Go to WORKSPACEDIR.
3. Download the JDK 21 (**jdk-<*version*>_linux-x64_bin.tar.gz**) for linux and copy it to **$WORKSPACEDIR/authorization-builder/staging/downloads/java**.
4. (Optional) If there is a change in the JDK version or in the file name, update the version and path accordingly in the manifest file as follows:

   ```
   $vi $WORKSPACEDIR/authorization-builder/bin/authorization_manifest.yaml

       jdk:
   ```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
    vendor: Oracle
    version: <JDK Version>
    path: $CN_BUILDER_STAGING/downloads/java/jdk-<version>_linux-
x64_bin.tar.gz
```

5. Set the proxy environment variables for running Gradle. For example:

```
#The eth0 is sample. replace "eth0" with your specific interface name.
export ip_addr=`ip -f inet addr show eth0|egrep inet|awk '{print $2}'|awk -
F/ '{print $1}'`
export http_proxy=
export https_proxy=$http_proxy
export no_proxy=localhost,$ip_addr
export HTTP_PROXY=
export HTTPS_PROXY=$HTTP_PROXY
export NO_PROXY=localhost,$ip_addr
```

6. Update **$WORKSPACEDIR/authorization-builder/bin/gradle.properties** with required proxies:

```
systemProp.http.proxyHost=
systemProp.http.proxyPort=
systemProp.https.proxyHost=
systemProp.https.proxyPort=
systemProp.http.nonProxyHosts=localhost|127.0.0.1
systemProp.https.nonProxyHosts=localhost|127.0.0.1
```

7. Run `build-all-images.sh` in **bin** to build all the images (authorization, authorization-schema):

```
cd $WORKSPACEDIR/authorization-builder/bin
./build-all-images.sh
```

> **Note**
>
> You can include the above procedure into your CI pipeline as long as the required components are already downloaded to the staging area.

Post-build Image Management

The Authorization image builder creates images with names and tags based on the settings in the manifest file. By default, this results in the following images:

- authorization-base-<*version*>:latest
- authorization-schema-<*version*>:latest
- authorization-<*version*>:latest

Copyright © 2023, 2026, Oracle and/or its affiliates.

Creating an Authorization Service Instance

This section describes how to create an Authorization service instance in your cloud native environment using the operational scripts and the configuration provided in the common cloud native toolkit.

Before you can create an Authorization service instance, you must validate cloud native environment. See "Planning UIM Installation" for details on prerequisites.

In this section, while creating a basic instance, the project name is considered as **sr** and instance name is considered as **quick**.

> **Note**
>
> Project and instance names cannot contain any special characters.

Installing Authorization Cloud Native Artifacts and Toolkit

Build container images for the following using the Authorization cloud native image builder:

- Authorization application
- Authorization schema installer

See "Deployment Toolkits" to download the Common cloud native toolkit archive file. Set the variable for the installation directory by running the following command,
where $WORKSPACEDIR is the installation directory of the COMMON cloud native toolkit:

```
export COMMON_CNTK=$WORKSPACEDIR/common-cntk
```

Setting up Environment Variables

The Authorization service relies on access to certain environment variables to run seamlessly. Ensure the following variables are set in your environment:

- Path to your common cloud native toolkit
- Path to your specification directory.

To set the environment variables:

1. Set the COMMON_CNTK variable to the path of directory where the Common cloud native toolkit is extracted as follows:

   ```
   $ export COMMON_CNTK=$WORKSPACEDIR/common-cntk
   ```

2. Set SPEC_PATH variable to the location where application and database YAML files are copied. See "Assembling the Specifications" to copy specification files if not already copied.

   ```
   $ export SPEC_PATH=$WORKSPACEDIR/spec_dir
   ```


Copyright © 2023, 2026, Oracle and/or its affiliates.

Creating Secrets

You must store sensitive data and credential information in the form of Kubernetes Secrets that the scripts and Helm charts in the toolkit consume. Managing secrets is out of the scope of the toolkit and must be implemented while adhering to your organization's corporate policies. Additionally, ATA service does not establish password policies.

Creating Secrets for Authorization Database Credentials

The database secret specifies the connectivity details and the credentials for connecting to the Authorization PDB (Authorization schema). This is consumed by the Authorization DB installer and Authorization runtime.

To create secrets for authorization database credentials:

1. Run the following script to create the required secrets:

```
$COMMON_CNTK/scripts/manage-app-credentials.sh -p <project-name> -i
<instance-name> -a authorization -s $SPEC_PATH create database

     <project-name> -> This value should match with the namespace in which
you want to run the helm install.
     <instance-name> -> This value should be matching with the instance
that you are going to specify while doing the helm install.
```

   For example:

```
$COMMON_CNTK/scripts/manage-app-credentials.sh -p sr -i quick -
s $SPEC_PATH -a authorization create database
```

2. Enter the corresponding values as prompted:
   - `Authorization DB Admin(sys) Username`: Provide Authorization Database admin username
   - `Authorization DB Admin(sys) Password`: Provide Authorization Database admin password
   - `Authorization Schema Username`: Provide username for Authorization schema to be created
   - `Authorization Schema Password`: Provide Authorization schema password
   - `Authorization DB Host`: Provide Authorization Database Hostname
   - `Authorization DB Port`: Provide Authorization Database Port
   - `Authorization DB Service Name`: Provide Authorization Database Service Name
3. Verify that the following secret is created:

```
$ kubectl get secret -n <namespace>
<project-name>-<instance-name>-authorization-db-credentials
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

Creating Secrets for Common Identity Provider Credentials

The OAuth (common identity provider) secret specifies the OIDC details of your identity provider. It is used by the Authorization service to protect the admin endpoints and for fetching the user profile information (subject, roles or groups, and so on) from access token. If authentication is enabled, ensure that you create an **oauthConfig** secret with the appropriate OIDC details of your identity provider.

To create secrets for common identity provider credentials, see "Adding Common OAuth Secret and ConfigMap" and "Common Configuration Options For all Services".

To provide your identity provider SSL certificate, create **commonTrust** secret that is mentioned in "Common TrustStore Secret".

Installing Authorization Service Schema

To install the Authorization service schema:

1. Update values under authorization-schema in **$SPEC_PATH/sr/quick/database.yaml** file with values required for authorization schema creation.

   > **Note**
   >
   > - The YAML formatting is case-sensitive. Use a YAML editor to ensure that you do not make any syntax errors while editing. Follow the indentation guidelines for YAML.
   > - Before changing the default values provided in the specification file, verify that they align with the values used during PDB creation. For example, the default tablespace name should match the value that you used while creating PDB.

2. Edit the **database.yaml** file and update the Authorization schema installer image to point to the location of your image as follows:

   ```
   authorization-schema:
     #imagePullPolicy: IfNotPresent
     schema:
       image:
         name: "authorization-schema-<version>"
         tag: latest

     db:
       defaultTablespace: SYSTEM
       tempTablespace: TEMP
   ```

3. If your environment requires a password to download the container images from your repository, create a Kubernetes secret with the Docker pull credentials. See "Kubernetes documentation" for details. Refer the secret name in the **database.yaml**. Provide the image pull secret and image pull policy details.

   ```
   authorization-schema:
     imagePullPolicy: IfNotPresent
   # The image pull access credentials for the "docker login" into Docker
   repository, as a Kubernetes secret.
   ```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
# Uncomment and set if required.
#  imagePullSecret: ""
```

4. Run the following script to start the Authorization Schema installer, which instantiates a Kubernetes pod resource. The pod resource lives until the Schema installation operation completes.

   ```
   $COMMON_CNTK/scripts/install-database.sh -p <project-name> -i <instance-
   name> -s $SPEC_PATH -a authorization -c 1

       <project-name> -> This value should match with the namespace in which
   you are running the helm install.
       <instance-name> -> This value should be matching with the instance
   that you specified while creating the secret.
   ```

   For Example:

   ```
   $COMMON_CNTK/scripts/install-database.sh -p sr -i quick -s $SPEC_PATH -a
   authorization -c 1
   ```

5. Check the console to see if the Schema installer is installed successfully.
6. If the installation has failed, run the following command to review the error message in the log:

   ```
   kubectl logs -n sr sr-quick-authorization-schema
   ```

7. Clear the failed pod by running the following command:

   ```
    helm uninstall sr-quick-authorization-schema -n sr
   ```

8. Run the **install-database** script again to install the Authorization schema installer.

Creating an Authorization Service Instance in Your Environment

To create an Authorization service instance in your environment using the scripts that are provided with the toolkit:

1. Run the following command to create an Authorization service instance:

   ```
   $COMMON_CNTK/scripts/create-applications.sh -p sr -i quick -s $SPEC_PATH -
   a authorization
   ```

   The create-applications script uses the helm chart located in **$COMMON_CNTK/charts/authorization-app** to create and deploy an authorization service.
2. If the scripts fail, run the following command to review the error message in the log:

   ```
   kubectl logs -n sr sr-quick-authorization
   ```

3. Clear the failed pod by running the following command:

   ```
   helm uninstall sr-quick-authorization -n sr
   ```

4. Fix the issues, if any, and run the script again to deploy the Authorization service.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Upgrading the Authorization Instance

Upgrading Authorization is required when there are updates made to **applications-base.yaml**, **app-authorization.yaml**, **<shape>/authorization.yaml**, or other configuration files.

To upgrade the Authorization instance:

1. Run the following command:

```
$COMMON_CNTK/scripts/upgrade-applications.sh -p sr -i quick -s $SPEC_PATH -
a authorization
```

2. After running the script, validate the Authorization service by running the `application-status` script.

Restarting the Authorization Instance

To restart the Authorization instance:

1. Run the following command to restart Authorization service:

```
$COMMON_CNTK/scripts/restart-applications.sh -p sr -i quick -s $SPEC_PATH -
a authorization
```

2. After running the script, validate the Authorization service by running the `application-status` script.

Deleting the Authorization Service Instance and Authorization Schema

To delete the Authorization instance:

1. Run the following command to delete the Authorization service instance:

```
$COMMON_CNTK/scripts/delete-applications.sh -p <project-name> -i <instance-
name> -s $SPEC_PATH -a authorization
```

2. Run the following script to delete the Authorization Schema installed, which instantiates a Kubernetes pod resource. The pod resource stays until the Schema deletion completes:

```
$COMMON_CNTK/scripts/install-database.sh -p <project-name> -i <instance-
name> -s $SPEC_PATH -a authorization -c 2
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

5

Deploying OpenSearch and OpenSearch Dashboard

This chapter describes how to deploy the OpenSearch service and OpenSearch dashboard.

Configuring OpenSearch and OpenSearch Dashboard Images

The Common CNTK has a sample that provides deployment instructions for OpenSearch on Kubernetes cluster using Helm charts. For more information, see https://opensearch.org/docs/latest/install-and-configure/install-opensearch/helm/

> **Note**
>
> You can install any OpenSearch available in the market. Oracle recommends you to use OCI OpenSearch or OpenSearch with the configuration parameters provided under samples directory in **common-cntk**.

Installing OpenSearch Helm Charts

To install OpenSearch Helm charts, run the following commands:

```
helm repo add opensearch https://opensearch-project.github.io/helm-charts/
helm repo update
helm search repo opensearch //you should see 2 charts opensearch/opensearch &
opensearch/opensearch-dashboard
```

Creating an OpenSearch and OpenSearch Dashboard Instance without SSL and Oauth Enablement

This section provides information on creating an OpenSearch and OpenSearch Dashboard instance without having SSL and Oauth enablement.

Setting up Environment Variables

OpenSearch and OpenSearch Dashboard rely on access to certain environment variables to run seamlessly across different environments. The OpenSearch charts are available at **$COMMON_CNTK/samples/charts/opensearch**.

The environment variables are configured in **$COMMON_CNTK/samples/charts/opensearch/scripts/env.sh**.

Configure **OPENSEARCH_CHART_VERSION**, **OPENSEARCH_DASHBOARD_CHART_VERSION**, and **OS_NAMESPACE** in **env.sh** script.

Copyright © 2023, 2026, Oracle and/or its affiliates.

To set up the environmental variables:

1. Set the OpenSearch namespace as follows:

   ```
   export OS_NAMESPACE=sr
   ```

2. Set the OpenSearch chart version as follows:

   ```
   export OPENSEARCH_CHART_VERSION=<chart version>
   ```

3. Set the OpenSearch dashboard chart version as follows:

   ```
   export OPENSEARCH_DASHBOARD_CHART_VERSION=<OpenSearch Dashboard chart
   version>
   ```

> **Note**
>
> For the corresponding OpenSearch chart versions, see Unified Inventory and Topology Microservices.

Installing OpenSearch

For OpenSearch configuration overrides, see "Assembling the Specifications". To override the default values, edit the **$SPEC_PATH/$OS_NAMESPACE/opensearch/os_board_values.yaml** and **$SPEC_PATH/$OS_NAMESPACE/opensearch/os_engine_values.yaml** files.

Install OpenSearch as follows:

```
$COMMON_CNTK/samples/charts/opensearch/scripts/create-opensearch.sh
```

To configure Persistence and Storage for OpenSearch, provide storage class name under `storageClass` as follows:

```
persistence:
  storageClass: <storageclass>
```

If you use a private or alternative Docker registry, provide the registry URL in the following command. For example: "myregistry.example.com":

```
global:
  dockerRegistry: ""
```

Installing OpenSearch Dashboard

Install the OpenSearch dashboard using the following command:

```
$COMMON_CNTK/samples/charts/opensearch/scripts/create-opensearch-dashboard.sh
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

Accessing the OpenSearch Dashboard Service

Access the OpenSearch dashboard using the node port of the OpenSearch dashboard service in the namespace.

```
export NODE_PORT=$(kubectl get --namespace $OPENSEARCH_NS -o
jsonpath="{.spec.ports[0].nodePort}" services os-board-opensearch-dashboards)
export NODE_IP=$(kubectl get nodes --namespace $OPENSEARCH_NS-o
jsonpath="{.items[0].status.addresses[0].address}")
echo http://$NODE_IP:$NODE_PORT
```

Installing FluentD

Update the **$COMMON_CNTK/samples/charts/fluentd/template/fluentd-config-map.yaml** file with OpenSearch details such as type, host, port, scheme, user, password, and ssl_verify as follows:

```
#Export the kubernetes namespace to be used for OpenSearch installation
helm install fluentd-logging $COMMON_CNTK/samples/charts/fluentd --
values $COMMON_CNTK/samples/charts/fluentd/values.yaml
```

Creating an OpenSearch and OpenSearch Dashboard Instance with SSL and Oauth Enablement

This section provides information on creating an OpenSearch and OpenSearch Dashboard Instance with SSL and Oauth enablement.

Setting up Environment Variables

To set up environmental variables, see Setting up Environment Variables.

Configuring config.yml

While configuring **$COMMON_CNTK/samples/charts/opensearch/config.yml** for OpenSearch with an Identity Provider (IdP), set up authentication parameters to enable seamless and secure integration.

To configure **config.yaml**:

1. Define the `openid_connect_idp` settings to include the discovery URL, client ID, and client secret of the IdP, which are essential for OpenID Connect (OIDC) authentication.
2. Specify the authorization and token endpoints provided by the IdP to handle user logins and configure role mapping to assign user roles based on the IdP attributes.

This setup allows OpenSearch to authenticate users through the IdP, easing access management while enhancing the security and compliance using:

- `openid_connect_url`: The OpenID configuration URL of IdP.
- `client_id` and `client_secret`: To securely authenticate OpenSearch with the Identity Provider (IdP) for OpenID Connect.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Creating Secrets in OpenSearch

To create secrets in OpenSearch:

1. Create a Kubernetes secret for the IdP trust certificate as follows:

   ```
   kubectl create secret generic <opensearch-namespace>-opensearch-idp-cert
   --from-file idp-chain-cert.pem=<IDP chain cert path> -n <opensearch-
   namespace>
   ```

2. Create a Kubernetes secret for OpenSearch IdP configuration after providing the required information in **config.yml** as follows:

   ```
    kubectl create secret generic <opensearch-namespace>-opensearch-idp-
   config-secret --from-file config.yml=<path/config.yml> -n <opensearch-
   namespace>
   ```

3. Configure `rolesMappingSecret` for OpenSearch roles mapping as follows:

   ```
    kubectl create secret generic <opensearch-namespace>-opensearch-
   rolesmapping-secret --from-file roles_mapping.yml=<path/role_mapping.yml> -
   n <opensearch-namespace>
   ```

4. Configure `internalUsersSecret` for OpenSearch internal users as follows:

   ```
   kubectl create secret generic <opensearch-namespace>-opensearch-
   internalusers-secret --from-file internal_users.yml=<path/
   internal_users.yml> -n <opensearch-namespace>
   ```

> **Note**
>
> The default username and password are used.

Implement Custom Certificates in OpenSearch

To implement custom certificates in OpenSearch:

1. Create the common certificates and common keys.
2. Create a secret from the common certificate and common key as follows:

   ```
    kubectl create secret generic <opensearch-namespace>-opensearch-keystore-
   cert --from-file=commoncert.pem=<path/commoncert.pem> --from-
   file=commonkey.pem=<path/commonkey.pem> -n sr
   ```

3. Provide the `secretName` under `secretMounts` in **os_engines_values.yml** as follows:

   ```
   - name: commoncertandkey
     secretName: <opensearch-namespace>-opensearch-keystore-cert
     path: /usr/share/opensearch/config/certs
   ```


Copyright © 2023, 2026, Oracle and/or its affiliates.

4. Provide the Distinguished Names (DNs) for authorized admin users in **plugins.security.authcz.admin_dn** and for trusted nodes in **plugins.security.nodes_dn** to control the access and secure node communication in OpenSearch.
5. Configure disabling of the installation of demo configuration settings that include demo users, roles, and certificates as follows:

```
env:
  name: DISABLE_INSTALL_DEMO_CONFIG
  value: true
```

6. Update **os_engine_values.yaml** to enable Security Plugin for OpenSearch. Comment the following section or set the value to **false**:

```
extraEnvs:
- name: DISABLE_SECURITY_PLUGIN
  value: "false"
```

Create an OpenSearch Instance

To create an OpenSearch instance:

1. To create an opensearch instance:

```
$COMMON_CNTK/samples/charts/opensearch/scripts/create-opensearch.sh
```

2. Run the `runsecurity-admin.sh` script to apply the changes that are made while implementing the custom certificates as follows:

```
$COMMON_CNTK/samples/charts/opensearch/scripts/runsecurity-admin.sh
```

Configuring opensearch_dashboards.yml for OpenSearch Dashboards Settings

The **opensearch_dashboards.yml** configuration file is essential for setting up and customizing the behavior of OpenSearch Dashboards. This file contains various settings that control how the dashboards interact with OpenSearch, including authentication, security, and connection parameters as follows:

- `opensearch.password`: The Basic Authentication password for accessing OpenSearch.
- `opensearch_security.openid.base_redirect_url`: The redirect URL for OpenSearch authentication.
- `opensearch_security.openid.connect_url`: The OpenID configuration URL of IdP.
- `opensearch_security.openid.client_id`: The client ID for OpenSearch authentication with the IdP.
- `opensearch_security.openid.client_secret`: The client secret to secure communication with the IdP.
- `opensearch_security.openid.scope`: The authentication scope.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Creating Secrets in OpenSearch Dashboard

To create secrets for OpenSearch Dashboard:

1. Create a Kubernetes Secret with the IdP trust Certificate and mount the secret in **os_board_values.yml** under **secretMounts** as follows:

   ```
   kubectl create secret generic <opensearch-namespace>-opensearch-idp-cert --
   from-file idp-chain-cert.pem=<Path/IDP chain cert path> -n <opensearch-
   namespace>
   ```

   > **Note**
   >
   > You may skip this step if you have created a Kubernetes secret for OpenSearch earlier.

2. Create a Kubernetes secret containing the OpenSearch Dashboards configuration and the corresponding location within the container:

   ```
   kubectl create secret generic <opensearch-namespace>-opensearch-dashboard-
   config --from-file=opensearch_dashboards.yml=<path/
   opensearch_dashboards.yml> -n <opensearch-namespace>
   ```

Setting up Ingress Controller for OpenSearch Dashboard

Using HAProxy as an Ingress Controller for OpenSearch Dashboards provides a highly configurable and stable solution for managing external access to your OpenSearch environment. By deploying HAProxy as the ingress, you can direct traffic to OpenSearch Dashboards using custom routing rules, including domain-based and path-based routing, allowing a better traffic control. You can use any Generic Ingress Controller that is available in the market. The following samples are for HAProxy.

The OpenSearch Dashboards integrate seamlessly with HAProxy ingress in Kubernetes environments to provide secure and scalable access. By leveraging the HAProxy ingress controller, administrators can share the Dashboards service externally, allowing users to interact with the OpenSearch data through a web-based interface.

To setup HAProxy for dashboard, configure below values in **os_board_values.yml**:

- `ingressClassName`: Provides the ingress class name value under the **ingress.ingressClassName** field.
- `secretName`: Defines the TLS secret that contains the SSL certificate for HTTPS.
- `tls.host`: Specifies the domain for accessing OpenSearch Dashboards over HTTPS.
- `hosts.host`: Specifies the domain for accessing OpenSearch Dashboards over HTTPS.

Run the following to create the secret:

```
kubectl create secret tls <project>-<instance>-opensearch-ingress-tls-cert-
secret --cert=<path/commoncert.pem> --key=<path/commonkey.pem> -n <os-
namespace>
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

Registering OpenSearch or OpenSearch Dashboard in Identity Provider

You must create Oauth 2.0 client in your IDP. You can use the same client created for ATA service. For more information, see "Registering ATA in Identity Provider".

In addition to the steps you perform while registering ATA in IdP, add the redirect URI: **https://<opensearch-instance>.<opensearch-namespace>.opensearch.uim.org:<LB-PORT>/auth/openid/login**

Add the following Post Logout Redirect URI:

```
https://<opensearch-instance>.<opensearch-namespace>.opensearch.uim.org:<LB-PORT>
```

Create OpenSearch Dashboard Instance

To create a OpenSearch dashboard instance, run the following command:

```
$COMMON_CNTK/samples/charts/opensearch/scripts/create-opensearch-dashboard.sh
```

Accessing the OpenSearch Dashboard

Access the OpenSearch dashboard URL that should redirect to the IdP login page. After you successfully log in, the OpenSearch dashboard appears.

```
Dashboard URI:  https://
<instance>.<project>.opensearch.<hostSuffix>:<ingressController-port>/app/
home#/
Example: https://quick.sr.opensearch.uim.org:30543/app/home#/
```

> **Note**
>
> Make sure you add the entry in the hosts file: `<k8s cluster ip or external loadbalancer ip> <instance>.<project>.opensearch.<hostSuffix>`

Upgrading the OpenSearch and OpenSearch Dashboard Service

This section provides information about upgrading the OpenSearch and OpenSearch dashboard services.

Upgrade OpenSearch Service

Upgrading OpenSearch is required when there are updates in the OpenSearch values yaml file.

To upgrade the OpenSearch instance:

```
$COMMON_CNTK/samples/charts/opensearch/scripts/upgrade-opensearch.sh
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

Upgrade OpenSearch Dashboard Service

Upgrading OpenSearch dashboard is required when there are updates in the OpenSearch dashboards values yaml file.

To upgrade OpenSearch dashboard service:

```
$COMMON_CNTK/samples/charts/opensearch/scripts/upgrade-opensearch-dashboard.sh
```

After running the script, verify the OpenSearch dashboard service by accessing it on a browser.

Deleting the OpenSearch and OpenSearch Dashboard Service

This section provides information on deleting the OpenSearch and OpenSearch dashboard services.

Deleting OpenSearch Service

To delete the OpenSearch instance:

```
$COMMON_CNTK/samples/charts/opensearch/scripts/delete-opensearch.sh
```

Deleting OpenSearch Dashboard Service

To delete the OpenSearch dashboard instance:

```
$COMMON_CNTK/samples/charts/opensearch/scripts/delete-opensearch-dashboard.sh
helm uninstall fluentd-logging --namespace=$OPENSEARCH_NS
```

Alternate Configuration Options for OpenSearch and OpenSearch Dashboard

This section provides alternate configuration options for OpenSearch and OpenSearch dashboard.

Changing the Default OpenSearch Basic Auth Password

To enhance security, update the default admin password in i**nternal_users.yml** by generating a hashed password using the OpenSearch hash script.

To change the default OpenSearch basic auth password:

1. Navigate to the OpenSearch Security plugin directory in opensearch pod as follows:

   ```
   cd /usr/share/opensearch/plugins/opensearch-security/tools
   ```

2. Run the following:

   ```
   ./hash.sh "<your-password>"
   ```


Copyright © 2023, 2026, Oracle and/or its affiliates.

3. Copy the hashed password and update **internal_users.yml**.
4. Set the same password in **opensearch_dashboards.yml**.
5. Replace the existing password hash with the new hashed value to secure the admin account.
6. To ensure compatibility, add the same password (in plain text) under **opensearch.password** in **opensearch_dashboards.yml**.
   This configuration allows OpenSearch Dashboards to authenticate with the updated credentials, aligning password settings across both files.

Creating Ingest Pipeline for OpenSearch

Login to OpenSearch Dashboard and navigate to **DevTools** on the top-right corner of the Home page and run the following to add the **lang_pipeline** in OpenSearch:

```
PUT _ingest/pipeline/lang_pipeline
{
  "description": "Pipeline to ensure documents have a lang field",
  "processors": [
    {
      "script": {
        "lang": "painless",
        "source": """
          if (ctx.lang == null) {
            throw new IllegalArgumentException('Document does not have the
*lang* field');
          }
        """
      }
    }
  ]
}
```

Debugging and Troubleshooting

OpenSearch pods restarting with error, related to max virtual memory areas

When you get the error: **vm.max_map_count [65530] is too low, increase it to at least [262144]**. In OpenSearch pods, you need to set **vm.max_map_count** to **262144** on each worker node where the OpenSearch pod is getting scheduled. Run the following commands on the nodes:

```
sudo sysctl -w vm.max_map_count=262144

# Alternatively you can run the following to save changes permanently
echo "vm.max_map_count=262144" >> /etc/sysctl.conf
sysctl -p
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

After logging in to the OAuth-enabled OpenSearch dashboard, seeing user ID instead of username

If you get a user ID instead of a username in the user-info section, you can configure a custom property mapper to map the username to the **sub** claim of the access token and add it to your scope configured with the OAuth client.

In the case of Keycloak, you can create a user property mapper.

Copyright © 2023, 2026, Oracle and/or its affiliates.

6
Deploying SmartSearch

This chapter describes how to deploy the SmartSearch service.

About SmartSearch

SmartSearch is a micronaut application, when integrated with OpenSearch, offers a powerful, flexible, and feature-rich search experience that can be tailored to specific business and user needs. Using OpenSearch as the underlying engine, SmartSearch can handle large volumes of data, perform real-time indexing, and support complex querying to enhance search relevancy. Features such as autocomplete, fuzzy matching, synonym recognition, and intelligent ranking make it easier for users to locate precise information, even if search terms are partially matched or misspelled.

Setting up Environment Variables

The SmartSearch service relies on access to certain environment variables to run seamlessly. Ensure the following variables are set in your environment:

- Path to your common cloud native toolkit
- Path to your specification directory

To set the environment variables:

1. Set the COMMON_CNTK variable to the path of directory where common cloud native toolkit is extracted as follows:

   ```
   $ export COMMON_CNTK=$WORKSPACEDIR/common-cntk
   ```

2. Set SPEC_PATH variable to the location where applications, shape, configuration, and database yamls are copied. See "Assembling the Specifications" to copy specification files if not already copied.

   ```
   $ export SPEC_PATH=$WORKSPACEDIR/spec_dir
   ```

Creating Secrets

To create secrets:

1. Run the following command to generate the `openSearchCredentials` secret and enter the basic auth credentials of OpenSearch as prompted:

   ```
   $COMMON_CNTK/scripts/manage-app-credentials.sh -p sr -i quick -a
   smartsearch -s $SPEC_PATH create openSearchCredentials

    #Enter the values appropriately against  prompts:
    opensearch username: admin   #opensearch basic oauth credentials
    opensearch password: xxxxx
   ```


Copyright © 2023, 2026, Oracle and/or its affiliates.

2. Verify if the secrets are created as follows:

```
sr-quick-open-search-credentials
```

Secret for Authentication Details

The SmartSearch service uses configuration values from Kubernetes secret (`<namespace>-<instance>-oauth-credentials`) and Config Map (`<namespace>-<instance>-oauth-config-cm`) objects from the same namespace. This Secret and Configuration Map Kubernetes objects have to be created before deploying the SmartSearch service for authentication. See "Adding Common OAuth Secret and ConfigMap" for creating the secret.

Secret to Pass Egress Certificates to SmartSearch

You have to create a `commonTrust` secret to provide SSL certificates of IdP, OpenSearch, and so on to SmartSearch.

To create `commonTrust` secret:

1. Generate **truststore.jks** that contains the required certificates as follows:

```
#generate truststore with idp cert (applicable only if authentication is
enabled )
keytool -importcert -v -alias <param> -file <path to IDP cert file> -
keystore <truststorename>.jks -storepass <password>

#add opensearch ssl trust certificate in truststore
keytool -importcert -v -alias <param> -file <path to opensearch cert file>
-keystore <truststorename>.jks -storepass <password>
```

2. Run the below command to generate the Kubernetes secret. The system will prompt for the truststore file path and password of truststore. Provide the details accordingly.

```
$COMMON_CNTK/scripts/manage-app-credentials.sh -p $PROJECT -i $INSTANCE -
s $SPEC_PATH create commonTrust
```

Configuring the Application Specification

In **$SPEC_PATH/$PROJECT/$INSTANCE** update the specification files: **app-smartsearch.yaml** and **applications-base.yaml** to provide the values of required configurations such as SmartSearch image, tag and ingress controller configurations, and so on. The `image` tag can be updated when a new version of the image is released. This allows easy upgrades and ensures that the application runs the latest features and security patches.

To configure the application specification:

1. Download and `push` the SmartSearch image to your internal repository as follows:
   a. Download the **Oracle Communications Unified Inventory Management SmartSearch** image from Oracle Software Delivery Cloud.
   b. Load the image from tar file and `push` the image to a container repository. See "About Container Image Management" in *UIM Cloud Native Deploy Guide*.

```
podman load -i <smart-search-latest.tar>
podman tag smart-search:latest <conatiner repository>/smart-
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
search:latest
podman push <conatiner repository>/smart-search:latest
```

2. Edit **$SPEC_PATH/$PROJECT/$INSTANCE/applications-base.yaml** specification file as follows:

   a. Provide an image `pull` secret as follows to pull images from your private repository:

   ```
   # The image pull access credentials for Private repository, as a
   Kubernetes secret.
   # uncomment and set if required.

   # imagePullSecret:
   #    imagePullSecrets:
   #    - name: regcred
   ```

   b. Update `loadbalancerport` as follows:

   ```
   # For Generic Ingress Controllers:
   # If ssl is enabled this would be loadbalancer's ssl port.
   # IF ssl is disabled this would be loadblancer's non ssl port.
   # For examle ssl and non-ssl ports for external loadbalancer would be
   443 and 80 respectively.

   # If loadbalancer is not created, provide nodePort of HAProxy or any
   other Generic Ingress Controller

   loadBalancerPort: 30505
   ```

   c. For Generic ingress controller, update the following:

   - Update the following to provide the annotations to enable stickiness through cookies:

     ```
     # Valid values are GENERIC
     ingressController: "GENERIC"
     ```

   - Annotations for HAProxy is as follows:

     ```
     ingress:
       className: "haproxy"
       annotations:
          haproxy.org/cookie-persistence: "uimhaproxycookie"
     ```

   d. Verify the shape name provided in **applications-base.yaml** and the corresponding **$SPEC_PATH/$PROJECT/$INSTANCE/shapes/<shapeName>/smartsearch.yaml** file for resources. If you want to change shape according to `dev` or `prod`, you can change shape value as follows:

   ```
   shape: dev
   ```

3. Edit **$SPEC_PATH/$PROJECT/$INSTANCE/app-smartsearch.yaml** specification file as follows:

Copyright © 2023, 2026, Oracle and/or its affiliates.

a. Provide the name and tag of the SmartSearch image from your repository as follows:

```
smartSearch:
  image:
    name: <conatiner repository>/smart-search
      tag: 0.4.9
```

b. Set `pullPolicy` for SmartSearch image as follows:

```
smartSearch:
  imagePullPolicy: IfNotPresent
```

Creating a SmartSearch Instance

This section describes how to create a SmartSearch service instance in your cloud native environment using the operational scripts and the configuration provided in the Common cloud native toolkit (Common CNTK).

Before you create a SmartSearch instance, you must validate cloud native environment. See "Planning UIM Installation" for details on prerequisites.

In this section, while creating a basic instance, the project name is considered as `sr` and instance name is considered as `quick`.

> **Note**
>
> The project and instance names cannot contain any special characters.

To create a SmartSearch service instance in your environment using the scripts that are provided with the toolkit:

1. Run the following command to create an SmartSearch service instance:

   ```
   $COMMON_CNTK/scripts/create-applications.sh -p sr -i quick -s $SPEC_PATH -a
   smartsearch
   ```

   The `create-applications` script uses the Helm chart located in **$COMMON_CNTK/charts/smartsearch-app** to create and deploy a SmartSearch service.

2. Run the following command to create SmartSearch Ingress:

   ```
   $COMMON_CNTK/scripts/create-ingress.sh -p sr -i quick -s $SPEC_PATH -a
   smartsearch
   ```

   The **create-ingress** script creates an Ingress object that defines hostname and path-based routing rules to direct incoming requests to SmartSearch services through the Ingress Controller. It is registered with the Ingress Controller based on the **className** property specified in the **applications-base.yaml** file.

3. (Optional) If the scripts mentioned above fail, run the following command to review the error message in the log:

   ```
   kubectl logs -n sr sr-quick-smartsearch
   ```

4. Clear the failed pod by running the following command:

Copyright © 2023, 2026, Oracle and/or its affiliates.

```
helm uninstall sr-quick-smartsearch-n sr
```

5. Fix the issues if any and run the script again to deploy the SmartSearch service.

Accessing the SmartSearch Service

To access SmartSearch service, use the SmartSearch endpoint in the format: `http://<topology-instance>.<topology-project>.topology.<hostSuffix>:<port>/<appVersion>/<uri-endpoint>`

For example: SmartSearch API endpoint is `http://quick.sr.topology.uim.org:30505/20240801/health`

Make sure you have created SmartSearch ingress using **create-ingress** script before accessing the above URL.

> **Note**
>
> Make sure you add entry in **/etc/hosts**: `<k8s cluster ip or external loadbalancer ip> quick.sr.topology.uim.org`.
>
> For SmartSearch appVersion, see UIM Compatibility Matrix.

Creating SmartSearch Index and Metadata

The SmartSearch schema in OpenSearch is structured to optimize search performance and relevance, using six main indexes: Location, Logical Device, Physical Device, Equipment, Communication, and Events. Each index is defined by mappings that specify field types such as text for full-text search and keyword for exact matches and may include custom analyzers for handling text processing tasks such as tokenization and synonyms. Additionally, metadata is configured for each index to control search behavior, supporting features such as filters, sorting, and autocomplete to enhance user experience. All indexes are accessed using aliases, providing a flexible way to query and manage data. This setup ensures fast, accurate, and relevant search results tailored to user needs.

Prerequisites

Ensure the following environment variables are set:

- `SMARTSEARCHDOMAIN`: Set this to the SmartSearch service endpoint (format: `export SMARTSEARCHDOMAIN=<host:port/appVersion>`), using the corresponding version .

  > **Note**
  >
  > For SmartSearch appVersion, see UIM Compatibility Matrix.

- `CLIENT_SECRET`, `CLIENT_ID`, `SCOPE`, and `IDP_TOKEN_URL`: These are necessary for generating the access token for the SmartSearch service.
- Ensure `SMARTSEARCHDOMAIN` and `IDP_TOKEN_URL` are accessible and included in the `NO_PROXY` settings or that the proxy is disabled.
- Ensure the relevant domain entries are present in the **/etc/hosts** file. For example: `<instance>.<project>.topology.<hostSuffix>`

Copyright © 2023, 2026, Oracle and/or its affiliates.

Creating Index and Metadata

To create index and metadata:

1. Navigate to the **SmartSearch** directory:

   ```
   cd $COMMON_CNTK/scripts/smartsearch
   ```

2. Run the script to create indexes and metadata for NPD:

   ```
   ./refreshIndexes.sh --ssl-cert-smartsearch <path-to-cert> --ssl-cert-oauth
   <path-to-cert>
   ```

3. Run the following command to create indexes and metadata for Service Impact Analysis related indexes:

   ```
   ./indexConfig.sh --index smartsearch-event smartsearch-impact smartsearch-
   rejectedevent --action createIndex --ssl-cert-smartsearch <path-to-cert> --
   ssl-cert-oauth <path-to-cert>
   ./indexConfig.sh --index smartsearch-event smartsearch-impact smartsearch-
   rejectedevent --action updateMetadata --ssl-cert-smartsearch <path-to-
   cert> --ssl-cert-oauth <path-to-cert>
   ```

4. (Optional) For non-secure deployments, use the `--security-enable` false flag.

Validating the Indexes before Creating

To validate the indexes before creating:

1. Access the OpenSearch dashboard and navigate to **DevTools**.
2. Run the following command to verify if the required indexes are created:

   ```
   GET /_cat/indices
   ```

3. Check if the following indexes with any suffixes are present:
   - `smartsearch-location-{*}`
   - `smartsearch-communication-{*}`
   - `smartsearch-physicaldevice-{*}`
   - `smartsearch-logicaldevice-{*}`
   - `smartsearch-equipment-{*}`
   - `smartsearch-event-{*}`
   - `smartsearch-impact-{*}`
   - `smartsearch-rejectedevent-{*}`

   > **Note**
   >
   > If any of these indexes are missing, rerun the script.

4. If any of these indexes are already present, skip running the creation scripts for such indexes.

Copyright © 2023, 2026, Oracle and/or its affiliates.

> **Note**
>
> In case you already have the `smartsearch-event-{*} index`. Please check if you have any data located in that index by running following query in the OpenSearch **DevTools**:
>
> ```
> GET /smartsearch-event/_count
> ```

5. If the count returned is **0**, do not include the existing index by running the following in the **$COMMON_CNTK/scripts/smartsearch** directory. For non-secure deployments, use the `--security-enable false` flag.

   ```
   ./indexConfig.sh --index smartsearch-event --action deleteIndex --ssl-cert-smartsearch <path-to-cert> --ssl-cert-oauth <path-to-cert>
   ```

Validating the Indexes after Creating:

1. Access the OpenSearch dashboard and navigate to **DevTools**.
2. Run the following command to verify if the required indexes are created:

   ```
   GET /_cat/indices
   ```

3. Check if the following indexes with any suffixes are present:
   - `smartsearch-event-{*}`
   - `smartsearch-impact-{*}`
   - `smartsearch-rejectedevent-{*}`

> **Note**
>
> If any of these indexes are missing, rerun the script.

Creating Index and Metadata Required for Service Impact Analysis

Event and Impact Analysis service uses following indexes:

- `smartsearch-event`
- `smartsearch-impact`
- `smartsearch-rejectedevent`

Validating the Indexes before Creating the Index and Metadata

To validate the indexes:

1. Access the OpenSearch dashboard and navigate to DevTools.

Copyright © 2023, 2026, Oracle and/or its affiliates.

2. Run the following command to verify if the required indexes are already present:

   ```
   GET /_cat/indices
   ```

3. Check if the following indexes with any suffixes are present:

   - `smartsearch-event-{*}`
   - `smartsearch-impact-{*}`
   - `smartsearch-rejectedevent-{*}`

4. If any of the indexes exists, skip running the create scripts for those indexes.

> **Note**
>
> Existing Event Index Action:
>
> - In case you already have the `smartsearch-event-{*}` index. Check if you have any data located in that index by running the following command in the OpenSearch DevTools:
>
>   ```
>   GET /smartsearch-event/_count
>   ```
>
> - If the count returned is **0**, you can remove this existing index by running the following command in **$COMMON_CNTK/scripts/smartsearch**. For non-secure deployments, use the `--security-enable` **false** flag.
>
>   ```
>   ./indexConfig.sh --index smartsearch-event --action deleteIndex
>   --ssl-cert-smartsearch <path-to-cert> --ssl-cert-oauth <path-to-
>   cert>
>   ```
>
> - If the count is not 0, do not proceed with the index creation and contact Oracle Support.

Upgrading the SmartSearch Service

Upgrading SmartSearch is required when there are updates made to **applications-base.yaml**, **app-smartsearch.yaml**, or other configuration files.

To upgrade the SmartSearch instance:

1. Run the following command to upgrade the SmartSearch service:

   ```
   $COMMON_CNTK/scripts/upgrade-applications.sh -p sr -i quick -s $SPEC_PATH -a
   smartsearch
   ```

2. After script is run, validate the SmartSearch service by running the `application-status` script.

Copyright © 2023, 2026, Oracle and/or its affiliates.

3. If there are any changes in Ingress configurations, loadbalancer port, annotation, and so on, delete and Create SmartSearch Ingress as follows:

```
$COMMON_CNTK/scripts/delete-ingress.sh -p sr -i quick -s $SPEC_PATH -a
smartsearch

$COMMON_CNTK/scripts/create-ingress.sh -p sr -i quick -s $SPEC_PATH -a
smartsearch
```

Validating the SmartSearch Instance

To validate the SmartSearch instance, run the following to the check the status of the deployed SmartSearch instance:

```
$COMMON_CNTK/scripts/application-status.sh -p sr -i quick -s $SPEC_PATH -a
smartsearch
```

This script returns the status of the SmartSearch service deployments and pods status.

Monitoring the SmartSearch Health

Use the following command to monitor the SmartSearch endpoint health:

```
https://<loadbalancerhost>:<loadbalancerport>/<appVersion>/health
```

Restarting the SmartSearch Instance

To restart the SmartSearch instance:

1. Run the following command to restart the SmartSearch service:

   ```
   $COMMON_CNTK/scripts/restart-applications.sh -p sr -i quick -s $SPEC_PATH -a
   smartsearch
   ```

2. After running the script, validate the SmartSearch service by running the `application-status` script.

Deleting the SmartSearch Service

To delete the SmartSearch instance, run the following command to delete the SmartSearch service instance:

```
$COMMON_CNTK/scripts/delete-applications.sh -p <project-name> -i <instance-
name> -s $SPEC_PATH -a smartsearch
```

Alternate Configuration Options for SmartSearch

You can configure SmartSearch using the following alternate options.

Setting Up a Secure Communication using TLS

To set up a secure communication using TLS:

Copyright © 2023, 2026, Oracle and/or its affiliates.

1. Edit the **$SPEC_PATH/$PROJECT/$INSTANCE/applications-base.yaml** and set the TLS `enabled` to `true`.

   ```
   tls:
     # The enabled flag is to enable or disable the TLS support for the smart
   search m-s  end points
     enabled: true
   ```

2. Specify the SSL port of loadbalancer or Ingress Controller in **applications-base.yaml** as follows. You use this port to access the application from outside:

   ```
   loadbalancerport: 30543
   ```

3. Generate common certificates commoncert.pem and commonkey.pem. Refer "SSL Certificates" in Appendix.
4. Create IngressTLS secret to pass the generated certificate and key pem files:

   ```
   $COMMON_CNTK/scripts/manage-app-credentials.sh -p sr -i quick -
   s $SPEC_PATH -a smartsearch create ingressTLS
   ```

   The script prompts for the following detail:

   - Ingress TLS Certificate Path (PEM file): <path_to_cert.pem>
   - Ingress TLS Key file Path (PEM file): <path_to_key.pem>

5. Verify that the following secrets are created successfully:

   ```
   sr-quick-smart-search-ingress-tls-cert-secret
   ```

6. Create a SmartSearch Instance and Ingress as usual and access the SmartSearch endpoints using `hostname <instance>.<instance>.topology.uim.org` as follows:

   ```
   #If instance already running delete and create Ingress.
   #instance
   $COMMON_CNTK/scripts/upgrade-applications.sh -p sr -i quick -s $SPEC_PATH -
   a smartsearch

   $COMMON_CNTK/scripts/delete-ingress.sh -p sr -i quick -s $SPEC_PATH -a
   smartsearch
   $COMMON_CNTK/scripts/create-ingress.sh -p sr -i quick -s $SPEC_PATH -a
   smartsearch
   ```

7. Add entry in `/etc/hosts <k8s cluster ip or external loadbalancer ip> quick.sr.topology.uim.org`.
   The SmartSearch API endpoint is `https://quick.sr.topology.uim.org:<LB Port>/<appVersion>/<uri-endpoint>`

Enabling Authentication on SmartSearch

To enable Authentication on SmartSearch, create the OATUH 2.0 client on your IDP and use the same client with OpenSearch, ATA, Authorization and Message Bus Services. After creating the client, enable authentication on SmartSearch service as follows:

Copyright © 2023, 2026, Oracle and/or its affiliates.

1. Create `oauthConfig` secret with IDP OAUTH 2.0 client details. See "Adding Common OAuth Secret and ConfigMap" for information on creating `OauthConfig` secret.
2. Create `commonTrust` secret with **trustStore** file that contains IDP and OpenSearch SSL certificate. See "Common TrustStore Secret" for more information.
3. Set the **authentication.enabled** flag to `true` in **$SPEC_PATH/$PROJECT/$INSATNCE/applications-base.yaml** to enable authentication as follows:

   ```
   # The enabled flag is to enable or disable authentication
   authentication:
     enabled: true
   ```

4. Upgrade the SmartSearch service.

Supporting Wildcard Certificates

SmartSearch supports wildcard certificates. You can generate the wildcard certificates with the `hostSuffix` value provided in **applications-base.yaml**. The default is `uim.org`.

Change the `subDomainNameSeperator` value from period(.) to hyphen(-) so that incoming hostnames match the wildcard DNS pattern.

Make the following modifications to the **$SPEC_PATH/$PROJECT/$INSTANCE/applications-base.yaml** file.

Uncomment and provide the value of subDomainNameSeparator. The default is ".".

```
#Value can be changed as "-" to match wild-card pattern of ssl certificates.
#Example hostnames for "-" quick-sr-topology.uim.org
subDomainNameSeparator: "-"
```

Configure Logging for SmartSearch

To update **log** level for SmartSearch:

1. Update the **SmartSearch.env.LOGLEVEL** value with the required Log level (**ERROR**, **INFO**, **DEBUG**, **TRACE**) in **$SPEC_PATH/sr/quick/app-smartsearch.yaml**
2. Upgrade the SmartSearch instance as follows:

   ```
   $COMMON_CNTK/scripts/upgrade-applications.sh -p sr -i quick -s $SPEC_PATH -
   a smartsearch
   ```

Configuring Shape for SmartSearch

There are predefined shapes available: **devsmall**, **dev**, **prodsmall**, **prod**, and **prodlarge**. Review the corresponding configurations at location **$SPEC_PATH/$PROJECT/$INSTANCE/shapes/**. To use a specific shape, specify its name in the **applications-base.yaml** file.

You can also create and customize your own shapes. See "Customizing the Shapes" for more information. If you want to use a different shape for SmartSearch than for other services, specify the **shape** value in the **app-smartsearch.yaml** file.

Managing SmartSearch SSL Certificates

To renew SmartSearch ingress SSL certificates or to import any new Egress certificate in SmartSearch truststore, see "Managing Certificate Expiry ".

Copyright © 2023, 2026, Oracle and/or its affiliates.

Scaling Up or Down the SmartSearch Service

To scale up or down the SmartSearch service:

1. Provide the replica count in **shapes/<shape>/smartsearch.yaml** to scale up or down the SmartSearch pods. Make sure the same shape value is configured in **applications-base.yaml** or **app-smartsearch.yaml** in which you update the replicas.
2. Update **shapes/<shape>/smartsearch.yaml** to increase replica count to 3 for SmartSearch deployment as follows:

   ```
   smartSearch:
       replicas: 3
   ```

3. Apply the change in replica count to the running Helm release by running the upgrade-applications script as follows:

   ```
   $COMMON_CNTK/scripts/upgrade-applications.sh -p sr -i quick -s $SPEC_PATH -
   a smartsearch
   ```


Copyright © 2023, 2026, Oracle and/or its affiliates.

7

Deploying Unified Operations Message Bus

This chapter describes how to deploy Unified Operations Message Bus.

Unified Operations Message Bus Overview

The Oracle Communications Unified Operations Message Bus (OCUOMB) service is a distributed event store and stream-processing platform service. The Message Bus clients send or receive the events or messages to or from the Message Bus through a specific channel called **Topic**. This enables that the source and target clients or services are loosely coupled and asynchronous. Message Bus uses Apache Kafka in its platform to support the event store and stream-processing and for packaging. For deployment, Message Bus uses Strimzi.

Strimzi simplifies the process of running Apache Kafka in a Kubernetes cluster. Strimzi provides container images and operators for running Apache Kafka on Kubernetes. Strimzi operators are fundamental for the smooth running of Strimzi. These operators are software extensions to Kubernetes that make use of custom resources to manage applications and their components. These operators simplify the process of:

- Deploying, running, and upgrading the Kafka cluster and its components.
- Configuring and securing access to Kafka.
- Creating and managing Kafka topics.

Operators are the methods of packaging, deploying, and managing a Kubernetes application. The Strimzi operators extend Kubernetes functionality and automate common and complex tasks related to a Kafka deployment. By implementing knowledge of Kafka operations in code, Kafka administration tasks are simplified and require less manual intervention. See https://strimzi.io/docs/operators/latest/overview.html for more details on the Strimzi operators. Strimzi has the following operators:

- *Cluster Operator*: Deploys and manages the Apache Kafka clusters, Kafka Connect, Kafka Mirror Maker, Kafka Bridge, Kafka Exporter, Cruise Control, and the Entity Operator.
- *Entity Operator*: Comprises the Topic Operator and User Operator
- *Topic Operator*: Manages Kafka topics

See the following webssites for more information on Strimzi and Apache Kafka:

- Strimzi: https://strimzi.io/
- Apache Kafka: https://kafka.apache.org/

The Kafka Message Bus service provides scripts and helm charts to deploy and manage the Apache Kafka cluster in Kubernetes by using the Strimzi operator and Kubernetes Custom Resources definitions. The Message Bus service does not provide any image builder toolkits to build the container images and by default, Helm charts pull the required container images from the quay.io/strimzi container repository.

Copyright © 2023, 2026, Oracle and/or its affiliates.

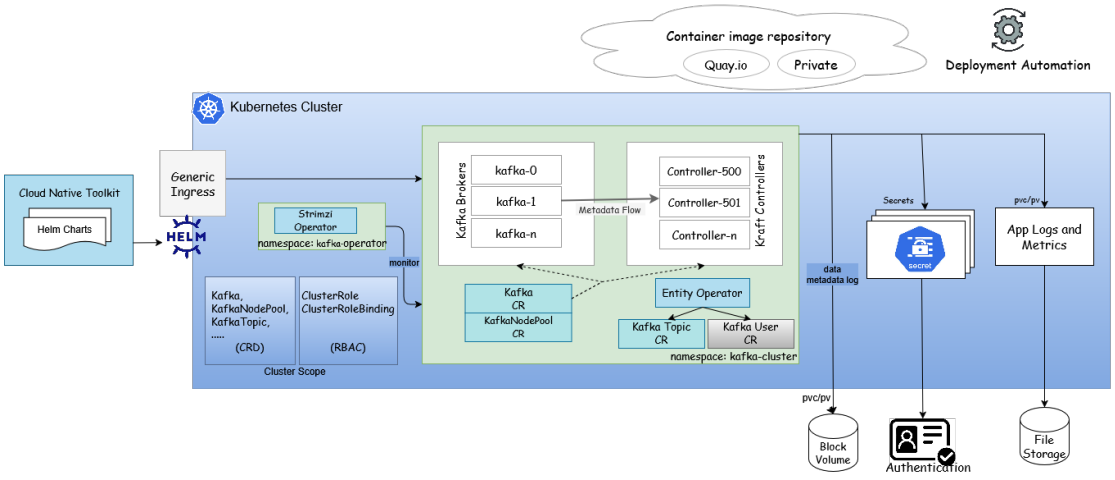
Table 7-1 Container Images and Purposes

Container Image	Purpose
quay.io/strimzi/operator:<*Strimzi_Operator_version*>	Container Image with Strimzi Operator.
quay.io/strimzi/kafka:<*Strimzi_Operator_version*>-kafka-<*Kafka_version*>	Container Image with Apache Kafka and Strimzi distribution. In the following sections, the reference to the container image is named as STRIMZI_KAFKA_IMAGE_NAME

> **Note**
>
> See "UIM Software Compatibility" in *UIM Compatibility Matrix* for the latest versions of software.

Message Bus Cloud Native Architecture

The Message Bus service uses Apache Kafka as a distributed event store platform. To run an Apache Kafka cluster on Kubernetes, the Message Bus service uses the Strimzi operator. Strimzi is an open-source project that provides container images and operators for running Apache Kafka on Kubernetes.



Access to Message Bus

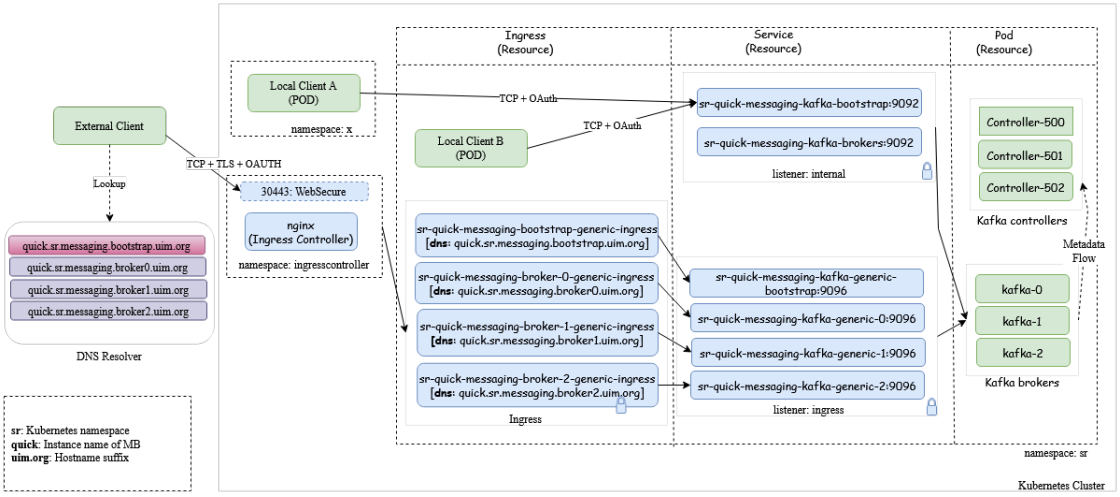
While deploying the Message Bus Service in Kubernetes namespace, the following Kubernetes service objects are created to access the Message Bus pods either internally or externally (through an ingress controller). The external access is provided through the ingress controller by IngressRouteTCP objects.

Copyright © 2023, 2026, Oracle and/or its affiliates.

> **Note**
>
> You can override the value of **subDomainNameSeparator**. The default separator is **"."**, This value can be modified as **"-"** to match the wild-card pattern of SSL certificates.

To override, uncomment and change the value in **applications-base.yaml** as follows:

```
#subDomainNameSeparator: "."
#Example hostnames for "-" : quick-sr-messaging-bootstrap.uim.org
```



The external access to Message Bus service is supported with **TCP+TLS+OAuth 2.0 Authentication** through an ingress controller. The internal access to Message Bus Service is also supported with **TCP+TLS+OAuth 2.0 Authentication** where TLS can be configurable. Access to Message Bus service is configured through the listeners section in **applications-base.yaml** and **app-messaging-bus.yaml** files.

> **Note**
>
> - If the client is in the same Kubernetes cluster, the **internal** listener is used.
> - If the client is outside the Kubernetes cluster, the **ingress** listener is used.

The Message Bus is deployed using the scripts provided in Common CNTK. For deployment prerequisites, see "Planning UIM Installation".

The following steps need to be followed to deploy a Kafka cluster in a Kubernetes namespace in a cluster:

1. Deploy the Strimzi operator to manage your Kafka cluster.

Copyright © 2023, 2026, Oracle and/or its affiliates.

> **Note**
>
> This is an administrative one-time activity where additional cluster roles are required.

a. Create a namespace to deploy Strimzi Operator.

b. Deploy Strimzi Operator in the namespace. See "Deploying Strimzi Operator" for more information.

2. Deploy the Message Bus that has Kafka brokers, Kafka controllers, and entity operator.

a. Create a namespace to deploy the Kafka cluster.

b. Register the namespace with Strimzi Operator. See "Register namespaces with Strimzi Operator" for more information.

> **Note**
>
> - The ingress controller has to be available.
> - Ensure that **ingress.className** is set in **applications-base.yaml**.

c. Deploy Kafka Cluster in the namespace. See "Deploy Kafka Cluster and Kafka Topic" for more information.

3. Validate the deployment with sample standalone producer and consumer clients. See the "Validating the Kafka cluster" and "Internal access - same namespace - plain" for more information.

Strimzi Operator

Export the Strimzi operator namespace environment variable to run the deployment script using the COMMON_CNTK:

```
export STRIMZI_NS=<STRIMZI_OPERATOR_NAMESPACE>
```

Configuration

The configurable parameters of the Strimzi Operator charts and their default values are listed in the corresponding subsections within this document.

To override the default values, copy the **$COMMON_CNTK/samples/strimzi-operator-override-values.yaml** file to the directory **$SPEC_PATH/<STRIMZI_PROJECT>**, where **<STRIMZI_PROJECT>** is the Kubernetes namespace where the Strimzi operator is being deployed. This file is copied to the **$SPEC_PATH/<STRIMZI_PROJECT>** location when you assemble the specifications. See "Unified Inventory and Topology Toolkit" for more information.

Create Global Resources

While deploying multiple Strimzi operators in the same Kubernetes cluster, ensure that one operator has **createGlobalResources** set to `true` in the **strimzi-operator-override-values.yaml** file, and all other operators are set to `false`. While other operators can use an earlier version of Strimzi Operator, Oracle recommends you to keep all operators on the same

Copyright © 2023, 2026, Oracle and/or its affiliates.

version to avoid potential compatibility issues with CRDs and the corresponding shared resources.

Private Container Repository

The Strimzi operator pulls the Strimzi component container images from **quay.io** registry. If you want to maintain private container registry, `pull` the images from the **quay.io** registry and `push` them into the private container registry. It is mandatory to push the images with same name and tag, the repository path can be different. For Strimzi image and tag names, see "Unified Operations Message Bus Overview" for more information.

See "About Container Image Management" section from *UIM Cloud Native Deployment Guide* for more information on private container repository management.

To use the private container registry, uncomment and modify the values in **$SPEC_PATH/<STRIMZI_PROJECT>/strimzi-operator-override-values.yaml** file. Provide the modified **strimzi-operator-override-values.yaml** file path as an `-f` option to the Strimzi operator create/upgrade command.

If the private container registry requires authentication, create the Kubernetes secret in the namespace and provide the secret name as part of **strimzi-operator-override-values.yaml** file. Create the secret with same name in the namespace where the Kafka cluster is planned to deploy.

strimzi-operator-override-values.yaml file (Sample)

```
defaultImageRegistry: <Image registry>
defaultImageRepository: <Image Repository>
image:
  imagePullSecrets: <Pull Secret>
```

The following is a sample command to create Kubernetes secret for the registry. Create the secret in the namespace where the Strimzi operator is being deployed. See https://kubernetes.io/docs/tasks/configure-pod-container/pull-image-private-registry/ for creating secret.

```
kubectl create secret docker-registry <secret-name> --docker-server=<Image
Registry> \
                                                    --docker-
username=<Username> \
                                                    --docker-
password=<Password> \
                                                    -n
<STRIMZI_OPERATOR_NAMESPACE>
```

ImagePullPolicy

The following sample of **ImagePullPolicy** for Strimzi Operator is provided. To create the policy using a different procedure, see https://kubernetes.io/docs/concepts/containers/images/#image-pull-policy

Copyright © 2023, 2026, Oracle and/or its affiliates.

strimzi-operator-override-values.yaml file (Sample)

```
image:
  imagePullPolicy: IfNotPresent
```

Resources

These resources are used for configuring the virtual resources (limits and requests). Uncomment or add the blow resources section with new values in the **strimzi-operator-override-values.yaml** file.

```
resources:
  requests:
    memory: <Mi>
    cpu: <m>
  limits:
    memory: <Gi>
    cpu: <"1">

fullReconciliationIntervalMs: 120000
operationTimeoutMs: 300000
```

The default values are as follows:

```
resources.limits.memory: 500Mi
resources.limits.cpu: 500m
resources.requests.memory: 1Gi
resources.requests.cpu: 1
```

Along with the above resources, you can provide the following additional configurations:

```
# Full reconciliation interval in milliseconds
fullReconciliationIntervalMs: 120000
# Operation timeout in milliseconds
operationTimeoutMs: 300000
```

Deploying Strimzi Operator

Run the following script to deploy the Strimzi operator in the Kubernetes namespace:

```
$COMMON_CNTK/scripts/strimzi-operator.sh -p <STRIMZI_OPERATOR_NAMESPACE> -c
create
```

Optionally, run the following script to deploy the Strimzi operator in Kubernetes namespace with custom image registry and repository:

```
$COMMON_CNTK/scripts/strimzi-operator.sh -p <STRIMZI_OPERATOR_NAMESPACE> -c
create -f $SPEC_PATH/<STRIMZI_OPERATOR_NAMESPACE>/strimzi-operator-override-
values.yaml
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

Upgrading Strimzi Operator

Run the following script to upgrade the Strimzi Operator in Kubernetes namespace:

```
$COMMON_CNTK/scripts/strimzi-operator.sh -p <STRIMZI_OPERATOR_NAMESPACE> -c
upgrade
```

> **Note**
>
> While upgrading **strimzi-cluster-operator** to a newer version, use the old toolkit for the older version of strimzi (the one already deployed) and new toolkit for upgrading to a newer version, in the case of **Create**, **Upgrade**, **Delete**, **Register** and **Unregister** operations.

Optionally, run the following script to deploy the Strimzi operator in Kubernetes namespace with custom image registry and repository:

```
$COMMON_CNTK/scripts/strimzi-operator.sh -p <STRIMZI_OPERATOR_NAMESPACE> -c
upgrade -f $SPEC_PATH/<STRIMZI_OPERATOR_NAMESPACE>/strimzi-operator-override-
values.yaml
```

Uninstalling Strimzi Operator

Run the following script to uninstall the Strimzi Operator from Kubernetes namespace:

```
$COMMON_CNTK/scripts/strimzi-operator.sh -p <STRIMZI_OPERATOR_NAMESPACE> -c
delete
```

Validating Strimzi Operator

Validate the Strimzi operator that is installed in the provided namespace by running the following command:

```
$kubectl get pod -n <STRIMZI_OPERATOR_NAMESPACE>

NAME                                        READY   STATUS    RESTARTS   AGE
strimzi-cluster-operator-*******-***        1/1     Running   0          6m55s
```

Validate the Helm release installed for the Strimzi operator in the provided namespace by running the following command:

```
$helm list -n <STRIMZI_OPERATOR_NAMESPACE>

NAME                    NAMESPACE                       REVISION
STATUS          CHART                         APP VERSION
strimzi-operator      <STRIMZI_OPERATOR_NAMESPACE>         1
deployed        strimzi-kafka-operator-x.y.z  x.y.z
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

Restarting the Strimzi Operator

Run the following script to restart the Strimzi Operator:

```
$COMMON_CNTK/scripts/strimzi-operator.sh -p <STRIMZI_OPERATOR_NAMESPACE> -c
restart
```

Registering the Namespaces with Strimzi Operator

To create and manage the Kafka cluster in a Kubernetes namespace, this namespace must be registered with the Strimzi operator to monitor the CRDs.

Run the following script to register the namespace(s) with the Strimzi operator to monitor and create or manage the Kafka cluster and its components:

```
$COMMON_CNTK/scripts/register-namespace.sh -p <Namespace to be monitored> -t
strimzi
```

Unregistering the Namespaces with Strimzi Operator

Run the following script to unregister the namespaces from the Strimzi operator:

```
$COMMON_CNTK/scripts/unregister-namespace.sh -p <Namespace to be un-
monitored> -t strimzi
```

Multiple Strimzi Operator

Strimzi Operator deployment deploys the cluster-wide resources such as Custom Resources Definitions (CRDs), Cluster Roles, Cluster Role Bindings, and so on. While deploying multiple operators you may come across conflicts or back-ward compatibility issues in these cluster resources.

The Strimzi operator team does not recommend you to deploy more than one Strimzi operator in a single Kubernetes cluster. If you have to deploy multiple operators to support different Kafka Message Services in cluster, ensure the following:

- Avoid deploying an older version of Strimzi operator when there is a new version.
- One operator should be deployed with create global resources as **true** and the sub-sequent operators must be deployed with **createGlobalResources** as **false**. Set this value in the input yaml file while deploying the operator.

> **Note**
>
> Ignoring these precautions can result in unpredictable behavior, conflicting updates to Kafka resources, and may result in potential failures in Kafka cluster management.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Configuring the Application Specification Files

Modify the values in the **applications-base.yaml** and **app-messaging-bus.yaml** files and upgrade or create the Message Bus service. The following configurations are available for the Message Bus service:

- Image Pull Secrets
- Security Context
- Cluster Size
- Storage
- Broker Defaults
- JVM Options
- Kafka Topics
- Accessing Kafka Cluster
- Authentication

Using Image Pull Secrets

You use the Image Pull Secrets sample only while using the private container repository that requires authentication. These authentication details have to be provided as Kubernetes secret object in the namespace where the Kafka cluster is planned to be deployed. This process is also followed while deploying Strimzi Operator.

> **Note**
>
> Provide the secret name in the **kafka-cluster** section, if using different secret name than in the Strimzi Operator's namespace.

Image Pull Secrets (Sample)

```
imagePullSecret:
  imagePullSecrets:
    - name: <secret name>
```

The sample command to create secret object for registry authentication is as follows:

```
kubectl create secret docker-registry <secret-name> --docker-server=<Image
Registry> \
                                                    --docker-
username=<Username> \
                                                    --docker-
password=<Password> \
                                                     -n <Kafka-Namespace>
```

See https://kubernetes.io/docs/tasks/configure-pod-container/pull-image-private-registry/ to create the secret object.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Security Context

The **userSecurity** section that has **securityContext** is applicable only when you want to define privilege and access control settings for a pod or container. The pod security context which is configured at the pod-level is provided as a sample and is applied to all containers in given pod.

> **Note**
>
> If a value is commented, it cannot be used, To use a different key-value, uncomment the corresponding value in **app-messaging-bus.yaml**.

See https://strimzi.io/blog/2022/09/09/configuring-security-context-in-pods-managed-by-strimzi/ and https://kubernetes.io/docs/tasks/configure-pod-container/security-context/ for more information.

Security-Context (Sample)

```
userSecurity:
  securityContext:
    runAsNonRoot: <true/false>
    runAsUser: <userID>
    runAsGroup: <groupID>
    fsGroup: <fsGroup>
```

Cluster Size

The Message Bus cluster consists of Kafka Brokers and Kafka Controller pods. Modify the replicas count for the Kafka Brokers and Controller pods according to the usage in **$SPEC_PATH/$PROJECT/$INSTANCE/<shape>/messaging-bus.yaml** and use the shape value configured in **applications-base.yaml**. For high availability of Message Bus service, make sure the number of replicas is minimum **3** for Kafka Broker and Controller, in production instance and adjust Kafka Broker configuration accordingly:

```
kafka-cluster:
  replicas:
    kafka: 3
    controller: 3
```

Storage

The Message Bus uses Strimzi to deploy the Apache Kafka cluster in Kubernetes cluster. For Strimzi to work as required, an efficient data storage infrastructure is essential. Oracle recommends using a block storage as Strimzi is tested for using with block storage. For more information on data storage, see https://strimzi.io/docs/operators/latest/deploying#considerations-for-data-storage-str

The Message Bus Service stores the events (or messages) in block storage using the Kubernetes Persistent Volumes. Modify the values for class, size, and **isDeleteClaim** values in

Copyright © 2023, 2026, Oracle and/or its affiliates.

storage section under the Kafka cluster. The storage class must have dynamic persistent volume provision capability:

```
kafka-cluster:
  #storage:
    #When storage.type below is set as "persistent-claim", the storage class
name & size are mandatory to be set
    #type: persistent-claim
    #class: psrnfsn1
    #size: 1Gi
    #isDeleteClaim: false
```

For development to use ephemeral (that is, temporary container storage), do not change the values. These values must be commented for ephemeral.

Broker Defaults

The following configuration is applied when the Topics are auto created. Modify the following settings in the **kafkaConfig** section under the Kafka cluster accordingly:

```
kafka-cluster:
  kafkaConfig:
    #The default replication factor for automatically created topics
    defaultReplicationFactor: 2
    offsetsTopicReplicationFactor: 2
    transactionStateLogReplicationFactor: 2
    transactionStateLogMinIsr: 2
    minInsyncReplicas: 1
    logRetentionMinutes: 120
    logSegmentBytes: 104857600
    numPartitions: 3
```

The values for replicationFactors and minimum in-sync replicas must be entered according to the values entered in the Kafka Cluster. These values must be less than or equal to the Kafka Cluster replica values.

For more information on the values, see the Kafka documentation at: https://kafka.apache.org/081/documentation.html#brokerconfigs

Configure the following properties to control how long Kafka retains log data and the maximum size of each log segment file:

- `logRetentionMinutes` determines how long Kafka retains messages before they become eligible for deletion.
- `logSegmentBytes` specifies the maximum size of each log segment file before Kafka creates a new segment.

You can combine time-based retention (for example: minutes or hours) with size-based retention (for example: bytes or megabytes) to ensure log data is neither older than a certain period nor exceeds a specific size, that can be customized as per your use case and storage requirements.

Setting higher values for these properties increases disk space usage on brokers for the relevant topic. However, setting lower values makes the data available for a lesser time, meaning consumers that are offline for extended periods risk missing messages.

Copyright © 2023, 2026, Oracle and/or its affiliates.

These configurations can be set at both the broker and topic levels. Topic-level settings for `logRetentionMinutes` and `logSegmentBytes` override broker-level defaults. If these properties are not explicitly set at the topic level, Kafka applies the broker-level values.

JVM Options

The Kafka Message Bus cluster consists of Kafka Brokers and Controller Pods. Modify the jvmOptions for Kafka nodes according to the usage. See https://strimzi.io/docs/operators/latest/full/configuring.html#con-common-configuration-jvm-reference for more details.

Heap dump on OutOfMemoryError

Heap dumps can be configured to capture diagnostic data when an **OutOfMemoryError** occurs. This requires setting up persistent storage for the heap dumps.

To enable this feature, you need to configure a **Storage Volume** for storing the heap dumps:

- – Set `storageVolume.enabled` to **true**.
- – Specify a Persistent Volume Claim (PVC) using the `storageVolume.pvc` parameter.

Sample Configuration:

```
jvmOptions:
  kafka:
    -Xms: 1024m
    -Xmx: 1024m
#  # Enable the following options only when PVC mount is properly configured
    #-XX:
    #  "HeapDumpOnOutOfMemoryError": "true"  # Only enable this when using a
volume mount
    #  "HeapDumpPath": "/mnt/logMount/kafka-heapdump.hprof"  # Ensure this
path exists
    # javaSystemProperties:
    #   - name: <placeHolder>
    #     value: <value>

  zookeeper:
    -Xms: 1024m
    -Xmx: 1024m
#  # Enable the following options only when PVC mount is properly configured
    #-XX:
    #  "HeapDumpOnOutOfMemoryError": "true"  # Only enable this when using a
volume mount
    #  "HeapDumpPath": "/mnt/logMount/zookeeper-heapdump.hprof"  # Ensure
this path exists
    # javaSystemProperties:
    #   - name: <placeHolder>
    #     value: <value>
```

Kafka Topics

Add or update the Kafka Topics in the **app-messaging-bus.yaml** file in the **kafkaTopics** section which are required for the Message Bus service clients (producers or receivers).

Copyright © 2023, 2026, Oracle and/or its affiliates.

For example:

```
kafka-topic:
  #List of Kafka topics
  kafkaTopics:
    - name: <topic1>
      partitions: <no_partitions>
      replicas: <no_replicas>
      config:
        retention: 7200000
        segmentBytes: 104857600
```

The following topics are required for the ATA integration which are defined in the **app-messaging-bus.yaml** file within the Common CNTK samples. These topics are created during the deployment of Message Bus service using Common CNTK:

Table 7-2 Topic, producer, and consumer details.

Topic	Producer	Consumer	Additional Details
ora-uim-topology	UIM	ATA	See Unified Inventory Management System Administration Overview in *UIM System Administrator's Guide* for more details.
ora-alarm-topology	Assurance System	ATA	For more information, see "Deploying the Active Topology Automation Service"
ora-retry-topology	ATA	ATA	For more information, see "Deploying the Active Topology Automation Service"
ora-dlt-topology	ATA	ATA	For more information, see "Deploying the Active Topology Automation Service"
ora-sol005-lcm	Sol Adapter	External sol5 consumer	
ora-alarm-retry	ATA	ATA	For more information, see "Deploying the Active Topology Automation Service"
ora-alarm-dlt	ATA	ATA	For more information, see "Deploying the Active Topology Automation Service"
ora-retry-smartsearch	ATA	ATA	For more information, see "Deploying the Active Topology Automation Service"
ora-dlt-smartsearch	ATA	ATA	For more information, see "Deploying the Active Topology Automation Service"
ora-test-topic	Standalone Test Client	Standalone test client	

Copyright © 2023, 2026, Oracle and/or its affiliates.

> **Note**
>
> Do not use the default topics (ora-uim-topology, ora-fault-topology, ora-retry-topology and ora-dlt-topology) for a standalone testing. Use only the **ora-test-topic** to test the deployment of Message Bus service.

Accessing Kafka Cluster

There are various listener type configurations available to access the Message Bus service internally and externally. The Authentication configuration is applied across all listener types. As part of Kafka cluster deployment, the Kubernetes service objects are created to provide access to Kafka cluster pods. This service objects are created based on the listener type configuration in the **app-messaging-bus.yaml** file for message-bus section. You can access the Message Bus service in any of the following ways:

- Accessing within the same cluster (Internal access)
- Accessing from outside of the cluster (External access)

> **Note**
>
> When a Message Bus service is deployed, it autogenerates the certificates of TLS for server and client. You must use the custom certificates so that the certificates are retained when the service is terminated and created again.

Accessing the Message Bus service from within the same cluster (Internal access)

The **internal** listener configuration in the **app-messaging-bus.yaml** file is used when the client services are in the same Kubernetes cluster, which can be in the same namespace or a different namespace. This configuration is enabled by default.

```
kafka-cluster:
  listeners:
    #Plain is for internal access within the same k8s cluster.
    internal:
      # Enable the tls to true if encryption/decryption is needed for
internal access
      #tls: false
```

See "Configuring Message Bus Listeners" for more information.

Accessing the Message Bus service from outside of the cluster (External access)

The **ingress** listener configuration in the **applications-base.yaml** and **app-messaging-bus.yaml** files are used when the client services are outside of the Kubernetes cluster. This access is achieved using the ingress controller.

Edit **applications-base.yaml** as follows:

```
# To expose the kafka-cluster to external kafka clients via ingress
controller uncomment the following and modify accordingly.
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
# Valid values is GENERIC
ingressController: <INGRESS_CONTROLLER>

#ingress:
#  #specify className field for ingressClassName of generic ingress
controller.
#  #In case of haproxy the default values is haproxy
#  className: "haproxy"

#provide loadbalancer port
# if TLS is enabled in global section, then loadbalancerport will be used as
external port for Generic Ingress Controller.
loadbalancerport: <loadBalancer-port>
```

Edit **app-messaging-bus.yaml** as follows:

```
kafka-cluster:
  listeners:
    #To expose the kafka-cluster to external kafka clients via ingress
controller uncomment the following and modify accordingly.
    #ingress:
    #  #The secure port of either ingress controller or external load-
balancer. If TLS is Disabled in global, then below ingressSslPort will be
used as external port.
    #  ingressSslPort: <LOADBALANCER_PORT>
    #  #If using Generic Ingress controller, below given annotations are
mandatory for Message-Bus external access.
    #  #These annotations are required for haproxy ingress controller.
    #  annotations:
    #   haproxy.org/ssl-passthrough: "true"
    #   ingress.kubernetes.io/ssl-passthrough: "true"
```

See "Configuring Message Bus Listeners" for more information.

Accessing the Message Bus service using a nodeport listener

The **nodeport** listener configuration in **app-messaging-bus.yaml** file configuration is also used when the client services are outside of the Kubernetes cluster. The access is directly with the Kubernetes work node's port.

> **Note**
>
> Oracle does not recommend this listener for production. It must be used only for debugging where ingress controller is not deployed.

```
kafka-cluster:
  listeners:
    #To expose the kafka-cluster to external kafka clients without ingress
controller, uncomment the following section and modify accordingly
    #nodeport:
      #default is true. can be turned off if needed
      #tls: true
      #if need to expose on a static nodeport, pease uncomment the below
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
section and provide values
      #nodePort: 32100
```

See "Configuring Message Bus Listeners" for more information.

Configuring Authentication

Kafka 2.0.0 or later supports an extensible OAuth 2.0 compatible token-based mechanism available, called **SASL OAUTHBEARER**. Strimzi has developed extensions that provide integration with OAuth 2.0 compliant authorization servers. That means, in principle, you can use any OAuth 2.0 compliant authorization server to enable centrally managed users for authentication with Kafka.

The Message Bus service uses a Strimzi operator to deploy Kafka brokers and in-turn use OAuth 2.0 token-based authentication while establishing a session to a Kafka broker. With this authentication, Message Bus clients (or Kafka clients) and Kafka brokers communicate with a central OAuth 2.0 compliant authorization server. These Kafka clients use the authorization server to obtain access tokens and are configured with access tokens issued by the server. Kafka brokers communicate with authorization server to validate the tokens presented by the clients, thus confirming their identities. You can perform the validation of access token using a fast local JWT validation or a token validation using an **introspection endpoint**.

To configure OAuth 2.0 support for Kafka Brokers in the Message Bus service, you need to update **applications-base.yaml** and **app-messaging-bus.yaml** files and create or upgrade the service.

Prerequisites

- Configure the client for Kafka broker in the authorization server.
- Configure the clients for Kafka producer or consumer application in the authorization server.
- Kafka cluster is configured with **oauth** type Authentication. See the following sections.

Enable Authentication on Kafka Cluster:

This procedure describes how to configure Kafka brokers so that the broker listeners are enabled to use OAuth 2.0 authentication by using an authorization server.

> **Note**
>
> Oracle recommends to use OAuth 2.0 over an encrypted interface through a listener with **tls**. Plain listeners are not recommended.

To enable authentication on the Kafka cluster:

1. In **applications-base.yaml**, un-comment or set the **authentication.enabled** flag to **true** as follows:

   ```
   # The enabled flag is to enable or disable authentication authentication:
   enabled: true
   ```


Copyright © 2023, 2026, Oracle and/or its affiliates.

2. In **app-messaging-bus.yaml**, to use fast local JWT validation, set **useFastLocalJWTvalidation** value to **true** under **kafka-cluster.listeners.authentication**. If not set, the introspection endpoint is used for validation:

```
#Uncomment the below host aliases section and provide hostname to
ipaddresss mappings
#This will add entries to POD's /etc/hosts file for hostname resolution
when DNS and other options are not applicable.
#For more details see https://kubernetes.io/docs/tasks/network/customize-
hosts-file-for-pods/

#hostAliases:
#- ip: <ip-address>
  #hostnames:
  #- <hostname-1> # Ex. quick.sr.ohs.uim.org


#Sample sub-section for using fast local jwt validation
kafka-cluster:
  listeners:
    authentication:
      useFastLocalJWTvalidation: true
```

3. The Message Bus service uses other configuration values from Kubernetes Secret (**<namespace>-<instance>-oauth-credentials**) and Config Map (**<namespace>-<instance>-oauth-config-cm**) objects from the same namespace. This Secret and Configuration Map Kubernetes objects have to be created before deploying the Message Bus service for authentication. See "Adding Common OAuth Secret and ConfigMap" for creating the secret. The configuration values used are:
   - clientID: The client ID to identify the client.
   - clientSecret: The client secret used for authentication.
   - validIssuerUri: The URI of the token issuer used for authentication.
   - introspectionEndpointUri: The URI of the token introspection endpoint.
   - jwksEndpointUri: The endpoint with public keys of authentication server that has to be used for fast local JWT validation.
   - tlsTrustedCertificate: The trusted certificates for TLS connection to the authorization server.

The following optional values are supported for authentication. See Strimzi documentation https://strimzi.io/docs/operators/in-development/configuring.html#type-KafkaListenerAuthenticationOAuth-reference for details on each value. Add the following optional values as required, under the **kafka-cluster.listeners.authentication** section in **app-messaging-bus.yaml** file:

```
# Additional optional authentication values
kafka-cluster:
  listeners:
    authentication:
      oauthConfig:
        #Enable or disable audience checking
        checkAudience:
        #Enable or disable issuer checking.  By default issuer is checked
using the value configured by validIssuerUri
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
        checkIssuer:
        #The audience to use when making requests to the authorization
server's token endpoint
        clientAudience:
        #The scope to use when making requests to the authorization server's
token endpoint
        clientScope:
        #The connect timeout in seconds when connecting to authorization
server
        connectTimeoutSeconds:
        #Enable or disable TLS hostname verification. Default value is false.
        disableTlsHostnameVerification:
        #The read timeout in seconds when connecting to authorization server.
        readTimeoutSeconds:
        #URI of the User Info Endpoint to use as a fallback to obtaining the
user id
        userInfoEndpointUri:
        #Name of the claim from the JWT authentication token
        userNameClaim:
```

Using GC Logs

By default, GC logs are disabled, you can enable it and view the logs on `stdout` by using `kubectl logs <kafka-cluster-pod-name>`.

To Enable GC logs, update **$SPEC_PATH/$PROJECT/$INSTANCE/applications-base.yaml** file as follows:

1. Under `gcLogs` make `enabled` as `true`.
2. Uncomment the `gcLogs` option under `kafka-cluster` to override common values.

```
gcLogs:
  enabled: true
```

> **Note**
>
> You do not have to configure **fileSize** and **noOfFiles** as the logs are printed on the **stdout**.

Auto-Rebalancing on Cluster Scaling

When Apache Kafka cluster is scaled up (by adding new brokers) or scaled down (by removing brokers), the issue arises in the cluster having unbalanced topic partitions. On the other hand, scaling down may not be possible if the brokers to be removed host topic partitions. In both scenarios, rebalancing the cluster by using the Cruise Control integration within Strimzi is the solution.

Cruise Control is an open source system that supports the Apache Kafka operations: Monitoring the cluster workload, Rebalancing a cluster based on predefined constraints.

See https://strimzi.io/blog/2024/11/25/autorebalancing-on-scaling/ for more information and https://github.com/linkedin/cruise-control for provided features.

Copyright © 2023, 2026, Oracle and/or its affiliates.

To use the auto-rebalancing feature, make sure the following requirements are met in the Kafka Messaging Bus:

- The **cruiseControl** should be enabled in the YAML configuration file.
- At least 3 Kafka Broker replicas should be configured.
- Once enabled, you cannot scale down the replicas to less then 3; unless you disable the **cruiseControl**.

Enabling the Cruise Control for Auto-Rebalancing

You enable the Cruise Control for auto-rebalancing on cluster scaling feature in the input YAML file. The **cruiseControl** element should be nested under the **kafka-cluster** element in YAML file.

cruiseControl auto-rebalancing

```
####
# Uncomment or add the below element (under?) to enable the Cruise Control
and Auto Rebalance on Scale features
# Websecurity for cruise control should always be true.
####
  cruiseControl:
    enable: true
    websecurityEnabled: true
    # The autoRebalance sub-element, used for autoRebalance of topic
partitions across the brokers
    autoRebalance:
      # Set the addBrokers element to true to enable partition rebalance
after scale up.
      addBrokers: true
      # Set this removeBrokers element to true to enable partition rebalance
before scale down.
      removeBrokers: true
```

addBrokers Configuration

When new Kafka Brokers are added to the Kafka cluster, partitions must be redistributed to balance the load and maximize resource utilization. In such scenarios, the configuration helps with the following:

- When enabled, Cruise Control automatically moves partitions to new brokers as they join the cluster.
- To prevent newly added brokers from sitting idle.
- To ensure even distribution of traffic across all brokers.
- To avoid manual intervention for rebalancing.

removeBrokers Configuration

When brokers are removed (for example: for maintenance or cost optimization), partitions should be migrated to remaining brokers without disruption. In such scenarios, the configuration helps with the following

- When enabled, Cruise Control automatically migrates partitions from decommissioned brokers to active ones.

Copyright © 2023, 2026, Oracle and/or its affiliates.

- To prevent data loss and ensure message durability.
- To avoid traffic imbalance when a broker is decommissioned.
- To maintain stability during scaling down.

Deploying and Managing Message Bus

The Kafka cluster consists of Kafka Brokers and Controller pods. Once the Strimzi operator is successfully installed in the Kubernetes cluster and a namespace for the Kafka cluster is registered to monitor, you can deploy and manage the Kafka cluster.

Update the **applications-base.yaml** and **app-messaging-bus.yaml**files as per your requirement and verify the following configuration elements in the yaml file before deploying the Kafka cluster:

> **Note**
>
> If **applications-base.yaml** and **app-messaging-bus.yaml** are not copied to $SPEC_PATH/sr/quick, make sure you run **assemble-specification** script, where the **sr** is the Kubernetes namespace and **quick** is instance name.

- The Storage class name that is used to create persistent volumes.
- The Kafka cluster replicas, which is the number of Kafka Brokers and Kafka controller nodes.
- Virtual Resource sizing.
- The Kafka Broker default settings.
- The listeners to be exposed with authentication and TLS.
- Authentication details.
- Metrics enablement.
- Affinity settings
- Update partitions, replicas, and retention period values for the default Kafka Topics.

See "Configuring the Application Specification Files" for more details.

Deploying Message Bus

Run the following commands to deploy the Kafka cluster with Kafka Topics in a Kubernetes namespace:

```
$COMMON_CNTK/scripts/create-applications.sh \
-p <kafka cluster namespace> \
-i <kafka cluster instance name> \
-s <path to spec directory> \
-a messaging-bus
```

For example:

Copyright © 2023, 2026, Oracle and/or its affiliates.

In the following command, **sr** is a namespace and **quick** an instance name:

```
$COMMON_CNTK/scripts/create-applications.sh -p sr -i quick -s $SPEC_PATH -a
messaging-bus
```

Upgrading Message Bus

The Kafka cluster upgrade requires persistent storage enabled for rolling update. Oracle recommends you have multiple replicas so that the service is not down while upgrading.

Update the Kafka cluster configuration in the **applications-base.yaml**, **app-messaging-bus.yaml**, or **<shape>/messaging-bus.yaml** file:

```
$COMMON_CNTK/scripts/upgrade-applications.sh \
-p <kafka cluster namespace> \
-i <kafka cluster instance name> \
-s <path to spec directory> \
-a messaging-bus
```

For example, run the following command to upgrade the Kafka cluster and Kafka topic running in sr namespace with instance as quick:

```
$COMMON_CNTK/scripts/upgrade-applications.sh -p sr -i quick -s $SPEC_PATH -a
messaging-bus
```

Deleting Message Bus

Run the following script to delete or uninstall the Kafka cluster and Kafka Topic from the Kubernetes namespace:

```
$COMMON_CNTK/scripts/delete-applications.sh \
-p <kafka cluster namespace> \
-i <kafka cluster instance name> \
-s <path to spec directory> \
-a messaging-bus
```

For example: Run the following command to delete the Kafka cluster with Kafka topic running in sr namespace with instance as quick:

```
$COMMON_CNTK/scripts/delete-applications.sh -p sr -i quick -s $SPEC_PATH -a
messaging-bus
```

Validating Message Bus

As a part of the KRaft migration, Zookeeper pods and services are deprecated and will no longer be utilized. The Kafka Controller pods will replace the Zookeeper Pods, and only Kafka Services will remain active in the architecture.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Check the pods created for the Kafka cluster. The following sample output shows the internal listener configuration. If it has any external listener settings, the additional service objects appear:

```
$kubectl get svc -n sr

NAME                                    TYPE        CLUSTER-IP      EXTERNAL-
IP   PORT(S)                        AGE
sr-quick-messaging-kafka-bootstrap    ClusterIP   <ip address>
<none>        9091/TCP,9092/TCP            22m
sr-quick-messaging-kafka-brokers      ClusterIP   None
<none>        9090/TCP,9091/TCP,9092/TCP   22m
```

Check the Service object created for the Kafka cluster. The following sample output shows the Kafka Broker and Controller replica as **1**.

```
$kubectl get pod -n sr

 NAME                                                READY   STATUS
RESTARTS   AGE
 sr-quick-messaging-entity-operator-*****-****       1/1     Running
0          27h
 sr-quick-messaging-kafka-0                          1/1     Running
0          27h
 sr-quick-messaging-controller-500                   1/1     Running
0          27h
```

Check the Helm release:

```
$helm list -n sr

NAME                    NAMESPACE       REVISION        UPDATED
STATUS          CHART                   APP VERSION
sr-quick-messaging      sr              1               *****
deployed        kafka-cluster-<x.y.z>     <x.y.z>
```

Check the persistent volume claims created:

```
$kubectl get pvc -n sr`

NAME                                    STATUS
VOLUME                                    CAPACITY   ACCESS MODES
STORAGECLASS   AGE
data-sr-quick-messaging-kafka-0          Bound    pvc-0a153e05-df71-431e-9fca-
d2bb5b55c701  1Gi        RWO           sc              27h
data-sr-quick-messaging-controller-500   Bound    pvc-888e1926-5f5e-4541-
b3f7-2c4647b7cb6b  1Gi        RWO           sc              27h
```

Run a standalone producer or consumer. See "Internal access - same namespace – plain" to run standalone producer and consumer pods in a Kafka cluster namespace.

Copyright © 2023, 2026, Oracle and/or its affiliates.

> **Note**
>
> As part of deploying, upgrading, and deleting the Message Bus, the Kafka topics are also created, upgraded, and deleted from the configuration provided in the input yaml file.

Restarting Message Bus

The **restart-application.sh** script with application name as **messaging-bus** restarts all the subcomponents such as Kafka Broker, Controller, and Entity Operators of the Message Bus. Run the following command to restart:

```
$COMMON_CNTK/scripts/restart-applications.sh -p sr -i quick -s $SPEC_PATH -a
messaging-bus
```

> **Note**
>
> The Message Bus service restart requires to have multiple replicas so that the service is not down while upgrading and the replica count should be greater than or equal to **2**.

To validate the restart option, see "Validating Message Bus ".

Alternate Configuration Options

There are various alternate options for configuring the Message Bus.

Logging Configuration for Message Bus

Message Bus uses Apache log4j for logging. By default, the logging type is set to **inline** with the default `logLevel` set to **INFO**. You can modify these settings based on your requirements for debugging.

Log Types

The following types of logging configurations are available:

- Inline (default)
- External: The logs are directed to a configured Storage Volume, such as a Persistent Volume (PV) or an NFS share.

Configuring Storage Volume for Logging

To enable **external** logging, you must configure a Storage Volume to store logs persistently. While using the external logging:

- The `storageVolume.enabled` parameter must be set to **true**.

Copyright © 2023, 2026, Oracle and/or its affiliates.

- A Persistent Volume Claim (PVC) must be specified using the `storageVolume.pvc parameter`.

```
# The storage volume must specify the PVC to be used for persistent storage.
When enabled with PVC value, log data will be directed here
# Without PVC value, it will act like disabled only.
storageVolume:
  enabled: true
  pvc: kafka-nfs-pvc #Specify this only if case type is PVC
```

If PVC is not configured, the logging type should be set to **inline**. A sample configuration is as follows:

```
logging:
  conversionPattern: '%d{yyyy-MM-dd HH:mm:ss,SSS} %-5p [%X{userName}] [%t]
[%c{1}] %m%n'
  #maxFileSize: "10MB"
  #maxBackupIndex: 10
  kafka:
    # logType can be external or inline
    # When logType is set to external then the logs will be written to a PVC
mounted using storageVolume
    # external logType should only be enabled when storageVolume is enabled
and PVC is mounted
    logType: inline
    logLevel: INFO
  zookeeper:
    # logType can be external or inline
    # When logType is set to external then the logs will be written to a PVC
mounted using storageVolume
    # external logType should only be enabled when storageVolume is enabled
and PVC is mounted
    logType: inline
    logLevel: INFO
  entityOperator:
    # logType can be external or inline
    # When logType is set to external then the logs will be written to a PVC
mounted using storageVolume
    # external logType should only be enabled when storageVolume is enabled
and PVC is mounted
    logType: inline
    logLevel: INFO
```

Choosing Worker Nodes for Running Message Bus Service

Update the Message Bus service configuration section in the **app-messaging-bus.yaml** file to node affinity or pod affinity and anti-affinity to constrain which nodes your pod can be scheduled. Alternatively, co-locate the pods in same node (or separate) and run either create or upgrade script.

Node Affinity

Node affinity is conceptually similar to **nodeSelector**, that enables you to constrain which nodes your pod can be scheduled, based on the node labels.

Copyright © 2023, 2026, Oracle and/or its affiliates.

There are two types of node affinities:

- Schedule a pod using required node affinity: The scheduler cannot schedule the pod unless the rule is met.
- Schedule a pod using preferred node affinity: The scheduler tries to find a node that meets the rule. If a matching node is not available, the scheduler continues to schedule the pod.

Preferred node affinity

The sample configuration for enabling preferred node affinity is as follows:

```
kafka-cluster:
  affinity:
    nodeAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
          matchExpressions:
          - key: name
            operator: In
            values:
            - south_zone
```

Kubernetes pod is scheduled on the node with label name as *south_zone*. If node with label name: *south_zone* is not available, pod will still be scheduled on another node.

Pod Affinity and Anti-Affinity

The Pod Affinity or anti-affinity allows you to constrain which node your pod is eligible to be scheduled, based on the labels on other pods.

Similar to node affinity, there are two types of pod affinity and anti-affinity:

- `requiredDuringSchedulingIgnoredDuringExecution`
- `preferredDuringSchedulingIgnoredDuringExecution`

Pod Affinity

Assign a Kubernetes pod to a node based on the labels on other pods using the Pod Affinity in a Kubernetes cluster. Modify the Kafka cluster override values yaml file.

The sample configuration for enabling the **required** pod affinity is as follows:

```
kafka-cluster:
  affinity:
    podAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        - labelSelector:
            matchExpressions:
              - key: app.kubernetes.io/name
                operator: In
                values:
                  - kafka
          topologyKey: "kubernetes.io/hostname"
```

Kubernetes pod is scheduled on the node which contains a pod with label http://app.kubernetes.io/name: kafka.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Modify the Kafka cluster override values yaml file. The sample configuration for enabling the **preferred** pod affinity is as follows:

```
kafka-cluster:
  affinity:
    podAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
        - weight: 1
          podAffinityTerm:
            labelSelector:
              matchExpressions:
                - key: app.kubernetes.io/name
                  operator: In
                  values:
                  - kafka
            topologyKey: "kubernetes.io/hostname"
```

The Kubernetes pod is scheduled on the node which contains a pod with label http://app.kubernetes.io/name: kafka. If the node is not available, pod will still be scheduled on another node.

Pod anti-affinity

Assign a Kubernetes pod to a node based on the labels on other pods using pod anti affinity in a Kubernetes cluster.

Modify the Kafka cluster override values yaml file. The sample configuration with **required** pod anti-affinity is as follows:

```
kafka-cluster:
  affinity:
    podAntiAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        - labelSelector:
            matchExpressions:
              - key: app.kubernetes.io/name
                operator: In
                values:
                  - kafka
          topologyKey: "kubernetes.io/hostname"
```

Kubernetes pod is scheduled on the node which does not contain a pod with label http://app.kubernetes.io/name: kafka.

Modify the Kafka cluster's override values yaml file. The sample configuration with **preferred** pod anti-affinity is follows:

```
kafka-cluster:
  affinity:
    podAntiAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
        - weight: 1
          podAffinityTerm:
            labelSelector:
              matchExpressions:
                - key: app.kubernetes.io/name
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
            operator: In
            values:
            - kafka
      topologyKey: "kubernetes.io/hostname"
```

Kubernetes pod is scheduled on the node which does not contains a pod with label http://app.kubernetes.io/name: kafka. If node is not available, pod will still be scheduled on another node.

Configuring Shape for Message Bus

The predefined shapes: **devsmall**, **dev**, **prodsmall**, **prod**, and **prodlarge** are provided and the corresponding configurations are available at location **$SPEC_PATH/$PROJECT/$INSTANCE/shapes/**. To use a specific shape, specify the name in the **applications-base.yaml** file. You can also create and customize your own shapes. See "Customizing the Shapes" for detailed instructions. If you want to use a different shape for Messaging Bus, specify the **shape** value in the **app-messaging-bus.yaml** file.

Managing Message Bus Metrics

Metrics in Message Bus are configured by enabling the JMX Exporter and Kafka Exporter. JMX Exporter can be enabled to get JVM metrics of Kafka cluster and Kafka Exporter can be enabled on a Kafka cluster to extract additional Prometheus metrics data from Kafka brokers, which is related to offsets, consumer groups, consumer lag, and topics.

See https://strimzi.io/docs/operators/latest/overview.html#metrics-overview_str for more information on metrics from Strimzi.

Enable metrics

Enable Kafka Exporter and JMX Exporter in the **$SPEC_PATH/sr/quick/app-messaging-bus.yaml** file and upgrade or create the Message Bus service. The sample content is as follows:

```
kafka-cluster:
  metrics:
    kafkaExporter:
      enable: true
    jmxExporter:
      enable: true


# Add or Uncomment with below elements to generate the Cruise Control Merics
when it is enabled.
  cruiseControl:
    enable: true
    metrics:
      enable: true
```

The above configuration shows the Prometheus metrics for Kafka Brokers, Topics, Consumer Groups, and Cruise Control components on metrics end-point on the pods. You can view these details on Prometheus UI by configuring the Scrape job. You can view this information in the form of graphs using the Grafana dashboard

See https://github.com/danielqsj/kafka_exporter#metrics to see the exposed metrics.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Prometheus and Grafana setup

See Setting Up Prometheus and Grafana for more information.

Sample Grafana dashboards

Add the Prometheus data source and import the sample Grafana dashboards from Strimzi github.

The sample Grafana dashboard for Kafka Broker, Controller, Mirror Maker2, Cruise Control, and JMX exporters can be downloaded from the following links:

- JMX Exporter metrics: https://github.com/strimzi/strimzi-kafka-operator/blob/main/examples/metrics/grafana-dashboards/strimzi-kafka.json
- Kafka Exporter metrics: https://github.com/strimzi/strimzi-kafka-operator/blob/main/examples/metrics/grafana-dashboards/strimzi-kafka-exporter.json
- Kafka Controller (KRaft) metrics: https://github.com/strimzi/strimzi-kafka-operator/blob/main/examples/metrics/grafana-dashboards/strimzi-kraft.json
- Kafka Mirror Maker metrics: https://github.com/strimzi/strimzi-kafka-operator/blob/main/examples/metrics/grafana-dashboards/strimzi-kafka-mirror-maker-2.json
- Cruise Control metrics: https://github.com/strimzi/strimzi-kafka-operator/blob/main/examples/metrics/grafana-dashboards/strimzi-cruise-control.json

Configuring Metrics for Prometheus

See "Configuring Metrics for Services" for more information.

Metrics for Cruise Control

To configure metrics in Cruise Control, enable JMX Exporter that gets JVM metrics of a cruise control enabled cluster.

```
####
# Uncomment to enable Cruise Control Auto Rebalance on Scale
# Websecurity for cruise control should always be true.
# It needs to be set to false ONLY to use cruise-control-ui.
# Defaults to true
####
  cruiseControl:
    enable: false
    websecurityEnabled: false
    metrics:
      enable: false
```

You can visualize the metrics using the following:

- **Prometheus UI**: By configuring a Scrape job, you can view the metrics on the Prometheus interface.
- **Grafana Dashboard**: This tool provides graphical representations of the metrics, making it easier to monitor.

See "Managing Message Bus Metrics" for more information.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Client Access

Accessing Message Bus in events producer and consumers clients.

Internal Access in the Same namespace for Plain

When the message producer or consumer applications are in same namespace as the Message Bus service then they can access the Kafka cluster using the Bootstrap Kubernetes service object name and port.

Run the following command to test the standalone **producer**. Here the project namespace is **sr** and instance is **quick**.

```
$kubectl -n sr run kafka-producer-plain -ti \
--image=<STRIMZI_KAFKA_IMAGE_NAME> \
--rm=true --restart=Never \
-- bin/kafka-console-producer.sh \
--bootstrap-server sr-quick-messaging-kafka-bootstrap:9092 \
--topic ora-test-topic
```

Type a few lines of text and each ENTER sends a message to Kafka broker. Type **CTRL-C** to quit.

Run the following command to test the standalone **consumer**. Here the project namespace is **sr** and instance is **quick**.

```
$kubectl -n sr run kafka-consumer-plain -ti \
--image=<STRIMZI_KAFKA_IMAGE_NAME> \
--rm=true --restart=Never \
-- bin/kafka-console-consumer.sh \
--bootstrap-server sr-quick-messaging-kafka-bootstrap:9092 \
--group ora-uim-consumer-test --isolation-level read_committed \
--topic  ora-test-topic --from-beginning
```

You get responses after the validation is successful.

Internal Access in a Different namespace for Plain

When the massage producer or consumer applications are in different namespace than the Message Bus service then they can access the Kafka cluster using the bootstrap service name and port but need to suffix ***<namespace>.svc.cluster.local*** to the service name.

See "Internal access - same namespace - plain" section on running the standalone console test producer and consumer pods for testing. Replace the bootstrap-server url with **sr-quick-messaging-kafka-bootstrap.sr.svc.cluster.local**, where the namespace is **sr** and instance is **quick**.

Internal Access in the Same namespace for Authentication

When the message producer or consumer applications are in same namespace as the Message Bus service then they can access the Kafka cluster using the bootstrap Kubernetes service object name and port.

Create a test client pod definition.

Copyright © 2023, 2026, Oracle and/or its affiliates.

1. Copy the following YAML content into the bastion host (or worker node) as **mb-test-client-deployment.yaml** file.
2. Update the **hostAliases** section according to your OAuth service environment.
3. Update the **STRIMZI_KAFKA_IMAGE_NAME**.
4. Update the **OAUTH Endpoint**, **Client Id** and **Secret**.
5. Update the **OAUTH Endpoint**, **Client Id**, **Client Secret**, **Scope**, **Audience**, and anything else that are applicable to your client configuration.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name:  mb-test-auth-client-deployment
  labels:
    app:  mb-test-auth-client
spec:
  replicas: 1
  selector:
    matchLabels:
      app:  mb-test-auth-client
  template:
    metadata:
      labels:
        app:  mb-test-auth-client
    spec:
#     <Uncomment below and replace with your bootstrap and brokers DNS
names>
      #hostAliases:
      #- ip: <LOADBALANCER_IP>
      #  hostnames:
      #  - "<OHS_HOSTNAME>"
      containers:
      - name:  mb-test-client
        image: <STRIMZI_KAFKA_IMAGE_NAME>
        command:
        - "tail"
        - "-f"
        - "/dev/null"
        imagePullPolicy: IfNotPresent
        env:
        - name: OAUTH_TOKEN_ENDPOINT_URI
          value: <Update the OAUTH_TOKEN_ENDPOINT_URI>
        - name: OAUTH_CLIENT_ID
          value: <Update the OAUTH_CLIENT_ID>
        - name: OAUTH_CLIENT_SECRET
          value: <Update the OAUTH_CLIENT_SECRET>
       # - name: OAUTH_SCOPE
       #   value: <Uncomment and update OAUTH_SCOPE>
       # - name: OAUTH_AUDIENCE
       #   value: <Uncomment and update OAUTH_AUDIENCE>
        ports:
        - containerPort: 9090
          name: http
          protocol: TCP
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

Create the authentication properties in a file (mb_test_client.properties).

```
sasl.jaas.config=org.apache.kafka.common.security.oauthbearer.OAuthBearerLogin
Module required;
security.protocol=SASL_PLAINTEXT
sasl.mechanism=OAUTHBEARER
sasl.login.callback.handler.class=io.strimzi.kafka.oauth.client.JaasClientOaut
hLoginCallbackHandler
```

Run the test client container and provide authentication properties

```
#Apply the test client pod definition in the namespace (say "sr").
$kubectl apply -f mb-test-client-deployment.yaml -n sr

#Get the newly created pod name
$kubectl get pod -n sr | grep mb-test-auth-client-deployment

#Sample Output
#mb-test-auth-client-deployment-******-****         1/1     Running
0          98s

#Copy the mb_authentication.properties file into the pod
$kubectl -n sr cp mb_test_client.properties mb-test-auth-client-deployment-
******-****:/home/kafka/mb_test_client.properties
```

Test for message bus producer client:

- Start an interactive shell process in the test client pod.
- Export the environment variables needed for the authentication.
- Run the console producer command.
- Enter some string messages.

```
#Get the newly created pod name
kubectl get pod -n sr | grep mb-test-auth-client-deployment

#Exec into the newly created pod
kubectl exec -it mb-test-auth-client-deployment-******-**** -n sr -- bash

#Run the following test console producer
bin/kafka-console-producer.sh \
--producer.config /home/kafka/mb_test_client.properties \
--bootstrap-server sr-quick-messaging-kafka-bootstrap:9092 \
--topic ora-test-topic
```

Test for message bus consumer client:

- Start an interactive shell process in the test client pod.
- Export the environment variables needed for the authentication.
- Run the console consumer command.

Copyright © 2023, 2026, Oracle and/or its affiliates.

- You will see the previous string messages of producer.

```
#Get the newly created pod name
kubectl get pod -n sr | grep mb-test-auth-client-deployment

#Sample Output
#mb-test-auth-client-deployment-******-****         1/1     Running
0          98s

#Exec into the newly created pod
kubectl exec -it mb-test-auth-client-deployment-******-**** -n sr -- bash

#Run the following test console consumer
bin/kafka-console-consumer.sh \
--consumer.config /home/kafka/mb_test_client.properties \
--bootstrap-server sr-quick-messaging-kafka-bootstrap:9092 \
--topic ora-test-topic \
--from-beginning
```

External ingress access - SSL and Authentication

The external access to Message Bus is provided through Ingress controller with TLS enabled. The following must be performed in clients for testing:

- Export and import the Message Bus service (that is sr-quick-messaging-cluster-ca-cert, where sr is namespace and quick is instance) certificate into clients.
- Export and import the certificate of OAuth service into the clients.

> **Note**
>
> This is optional and is required only if OAuth is enabled for SSL.

- Update the bootstrap and brokers DNS names with load balancer IP in the **etc/hosts** file of clients (that is, event producer or consumer applications).
- Update the DNS name of OAuth service with load balancer IP in **/etc/hosts** file of clients.

> **Note**
>
> This is optional and is required only if the OAuth service requires DNS name to access.

- Run the producer or consumer script with SSL and Authentication details.

In the following section, the external ingress access test is provided with Strimzi Kafka container. If you want to test the client code without Kubernetes cluster then you can download the Apache Kafka and perform the same.

Add Message Bus service and OAuth service certifications to trust store. See **Import/export of TLS certificates** section.

```
#Run the below command to export and import the Message Bus service
certificate into the trust store (mb-cert-keystore.jks) file.
$COMMON_CNTK/scripts/export-message-bus-cert.sh -p sr -i quick -l . -k ./mb-
test-client-cert-keystore.jks -a mb-cert
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
#Get the OAuth IDP certificate and import into trust store (mb-test-client-
cert-keystore.jks) file (Optional, needed if OAuth is SSL)
keytool -importcert -alias oauth-server -file <Path to OAuth Server
certificate, the .pem file> -keystore ./mb-test-client-cert-keystore.jk --
trustcacerts -noprompt
```

Create the following authentication properties in a file (**mb_test_client.properties**).

```
sasl.jaas.config=org.apache.kafka.common.security.oauthbearer.OAuthBearerLogin
Module required;
security.protocol=SASL_SSL
sasl.mechanism=OAUTHBEARER
sasl.login.callback.handler.class=io.strimzi.kafka.oauth.client.JaasClientOaut
hLoginCallbackHandler
ssl.endpoint.identification.algorithm=
```

Create a test client pod definition.

1. Copy the following YAML content into the bastion host (or worker node) as "**mb-test-client-deployment.yaml**" file.
2. Update the Strimzi Kafka image.
3. Update the hostAliases section according to your OAuth and Message Bus service setup. This will add entries to /etc/hosts file.
4. Update the **OAuth Endpoint**, **Client Id**, **Client Secret**, and **Trust Store Password** in **env** section.

> **Note**
>
> You can override the value of **subDomainNameSeparator**. The default is **.**. This value can be changed as **"-"** to match the wild card pattern of SSL certificates.
>
> To override, uncomment and change this value in **applications-base.yaml**.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name:  mb-test-client-deployment
  labels:
    app: mb-test-client
spec:
  replicas: 1
  selector:
    matchLabels:
      app:  mb-test-client
  template:
    metadata:
      labels:
        app:  mb-test-client
    spec:
#     <Uncomment below and replace with your bootstrap and brokers DNS names>
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
#     hostAliases:
#     - ip: <Replace with your LOADBALANCER_IP>
#       hostnames:
#       - "<INSTANCE.PROJECT.messaging.broker0.uim.org>"
#       - "<INSTANCE.PROJECT.messaging.brokerN.uim.org>"
#       - "<INSTANCE.PROJECT.messaging.bootstrap.uim.org>"
#       - "<Replace with OHS_HOSTNAME>"
      containers:
      - name:  mb-test-client
        image: quay.io/strimzi/kafka:0.34.0-kafka-3.4.0
        command:
        - "tail"
        - "-f"
        - "/dev/null"
        imagePullPolicy: IfNotPresent
        env:
        - name: OAUTH_TOKEN_ENDPOINT_URI
          value: <Replace with your OAUTH_TOKEN_ENDPOINT_URI>
        - name: OAUTH_CLIENT_ID
          value: <Replace with your OAUTH_CLIENT_ID>
        - name: OAUTH_CLIENT_SECRET
          value: <Replace with your OAUTH_CLIENT_SECRET>
        #- name: OAUTH_SCOPE
        #  value: <Uncomment and replace with your OAUTH_SCOPE>
        #- name: OAUTH_AUDIENCE
        #  value: <Uncomment and replace with yours OAUTH_AUDIENCE>
        - name: KAFKA_OPTS
          value: " \
                  -Djavax.net.ssl.trustStore=/home/kafka/mb-test-client-cert-
keystore.jks \
                  -Djavax.net.ssl.trustStorePassword=<Replace with your store
password> \
                  -Djavax.net.ssl.trustStoreType=JKS"
        ports:
        - containerPort: 9090
          name: http
          protocol: TCP
```

Run the test client container and apply readiness for authentication and SSL.

```
#Apply the test client pod definition in the namespace (say "sr").
$kubectl apply -f mb-test-client-deployment.yaml -n sr

#Get the newly created pod name
$kubectl get pod -n sr | grep mb-test-client-deployment

#Sample Output
#mb-test-client-deployment-******-****         1/1     Running   0
98s

#Copy the certificate store into the newly created pod.  Replace the pod name
below
kubectl -n sr cp mb-test-client-cert-keystore.jks <Replace with mb-test-
client-deployment pod name>:/home/kafka/mb-test-client-cert-keystore.jks
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
#Copy the mb_test_client.properties file into the POD
kubectl -n sr cp mb_test_client.properties <Replace with mb-test-client-
deployment pod name>:/home/kafka/mb_test_client.properties
```

Start a shell session inside container for console **test producer**.

```
#Get the newly created pod name
$kubectl get pod -n sr | grep mb-test-auth-client-deployment

#Sample Output
#mb-test-auth-client-deployment-******-****         1/1     Running
0          98s

#Exec into the newly created pod
kubectl exec -it<Replace with mb-test-client-deployment pod name> -n sr --
bash

#Run the following producer command
bin/kafka-console-producer.sh \
--producer.config /home/kafka/mb_test_client.properties \
--bootstrap-server quick.sr.messaging.bootstrap.uim.org:30443 \
--topic ora-test-topic
```

Start start a shell session inside container for console **test consumer**:

```
#Exec into the newly created pod
kubectl exec -it <Replace with mb-test-client-deployment pod name> -n sr --
bash

#Run the following producer command.  Replace the bootstrap-server url
accordingly to your environment
bin/kafka-console-consumer.sh \
--consumer.config /home/kafka/mb_test_client.properties \
--bootstrap-server sthatipa.sr.messaging.bootstrap.uim.org:30443 \
--consumer-property group.id=test-client-service \
--topic ora-test-topic --from-beginning
```

Clean-up the newly created test pod:

```
kubectl delete -f mb-test-client-deployment.yaml -n sr
```

External node port access

The nodeport listener type allows the external access from outside of the Kubernetes cluster using the load balancer or Kubernetes worker node ip address and nodePort (port of worker node).

The Bootstrap URL is constructed with worker node IP Address and node port of bootstrap service.

Get the host port of the external bootstrap service using the following command:

```
$kubectl get service sr-quick-messaging-kafka-nodeport-bootstrap -
o=jsonpath='{.spec.ports[0].nodePort}{"\n"}' -n sr
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
Output: 32100
```

Get the IP Address of the Kubernetes worker node. Replace the <NODE_NAME> in the following with your node name:

```
$kubectl get node <NODE_NAME> -o=jsonpath='{range .status.addresses[*]}{.type}
{"\t"}{.address}{"\n"}' -n sr

Output:
InternalIP  100.xx.xx.142
Hostname    *********
```

Update the Kafka cluster Bootstrap URL as **100.xx.xx.142:32100** in the events producer and consumer applications.

To access with plain, see "Internal access - same namespace - plain" section. Replace the bootstrap URL with above constructed one.

To access with Authentication, see "Internal access - same namespace - authentication" section. Replace the bootstrap URL with above constructed one.

To access with SSL and Authentication, see "External ingress access - SSL & Authentication" section. Replace the bootstrap URL with above constructed one.

Import/export of TLS certificates

To enable TLS encrypted access, the ca-certs of Kafka cluster is needed to be extracted and imported into key store and the location of that key store is used as the producer or consumer properties in events application.

Export the ca-certs of the Kafka cluster using the following command:

```
$COMMON_CNTK/scripts/export-message-bus-cert.sh -p <Namespace of kafka
cluster> \
-i <instance name of kafka cluster> \
-l <directory to export clustercerts temporarily> \
-k <keystore-location> \
-a <alias for cert>
```

For example:

```
$COMMON_CNTK/scripts/export-message-bus-cert.sh -p sr -i quick -l . -k ./mb-
cert-keystore.jks -a mb-sr-quick-cert
```

The export-cluster-cert.sh script creates JKS type truststore by default in the provided key store location. If any other truststore type is created, specify that as producer or consumer property while running the clients. These exported artifacts can be used in Kafka client applications.

Copyright © 2023, 2026, Oracle and/or its affiliates.

> **Note**
>
> If custom certificates were used during cluster creation, then these can be directly provided through a keystore than extracting the generated certs.

Using custom certificates

Custom certificates can be used while creating the Kafka cluster:

Prerequisites:

- Certificates and keys are to be in PEM format.
- Key should not be encrypted. Encrypted keys are not supported since they need user interaction for entering the passphrase during access.

Creating a custom certificate

To create a custom certificate, see "SSL Certificates".

Create Kubernetes secret

Run the following command by replacing the placeholders:

```
kubectl create secret generic <secret-name> --from-file=<key-file-name> --
from-file=<certificate-file-name>
 For example:
kubectl create secret generic myCustomCertSecret --from-file=commonkey.pem --
from-file=commoncert.pem
```

Update Kafka Cluster configuration

Update the customCerts configuration section in Kafka cluster's override values yaml file:

```
kafka-cluster:
  ## to enable custom or owned certs for tls please create a kubernetes
secret with the cert and key if not already present, uncomment the below
section and add respective values.
    ## please be advised that encrypted keys are not supported since they
require user interaction for the passphrase
    customCerts:
      # Secret in which cert and key are present
      secretName: <secret-name created above>
      certName: <certificate file used in the secret created above>
      keyName: <key-file used in the secret created above>
```

Configuring Message Bus Listeners

Message Bus has three listeners (internal, ingress and nodeport) to access the service. These are described in the following sections.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Message Bus Internal Listener

The following is the configuration for **internal** listener type which can be commented or uncommented.

```
kafka-cluster:
 listeners:
 # plain is for internal access within the same k8s cluster.
   internal:
```

From same namespace in cluster

This is an internal access method that is used by the message producer or consumer clients (or applications) when they are deployed in same namespace as the Message Bus service. This is enabled by default with **internal** listener type. To access the Message Bus, the producer or consumer applications must get the Bootstrap service URL of the Kafka cluster.

To get the Bootstrap service URL of the Kafka cluster run the following command:

```
kubectl get svc -n sr | grep sr-quick-messaging-kafka-bootstrap

sr-quick-messaging-kafka-bootstrap       ClusterIP   <clusterIP>
<none>        9091/TCP,9092/TCP
```

> **Note**
>
> The project namespace is **sr** and instance is **quick**.

Use the **sr-quick-messaging-kafka-bootstrap:9092** URL in the producer and consumer client configuration in the applications.

From another namespace in cluster

This is an internal access method which is used by the producer or consumer client applications when they are deployed in different namespace than the message-bus service. This is enabled by default with internal listener type. To access the Message Bus, the producer or consumer client applications have to get the Bootstrap service URL of the Kafka cluster and convert the URL pattern as **serviceName.namespace.svc.cluster.local**.

If the *sr-quick-messaging-kafka-bootstrap* service is hosted in **sr** namespace on 9092 port and the client applications from different namespace can access the Kafka cluster with Bootstrap URL as ***sr-quick-messaging-kafka-bootstrap.sr.svc.cluster.local:9092***

Message Bus Ingress Listener

This is an external access method which is used by message producer or consumer applications when they are deployed outside of the Kubernetes cluster. This is disabled by default and must be enabled in the **app-messaging-bus.yaml**. This external access is provided through the Ingress Controller to the Kafka cluster. To enable this external access, the ingress listener type configuration must be enabled in the Kafka cluster configuration yaml file.

Copyright © 2023, 2026, Oracle and/or its affiliates.

> **Note**
>
> While deploying Ingress Controller, ensure that the controller supports SSL `passthrough` feature using annotations and is enabled.
>
> `--set "controller.extraArgs.enable-ssl-passthrough="`

Ingress listener type

Un-comment the ingress lister type section in **applications-base.yaml** file to expose the Message Bus Service outside of Kubernetes cluster. Ingress controller should be deployed in order for this ingress listener type to work. In case of Generic Ingress, set **ingress.className** according to your Generic Ingress Controller.

In case of Generic Ingress controller (HAProxy), annotations given under the **kafka-cluster.listeners.ingress.annotations** tag in **applications-base.yaml** are mandatory.

```
# To expose the kafka-cluster to external kafka clients via ingress
controller uncomment the following and modify accordingly.
ingressController: "GENERIC"

#ingress:
#  #specify className field for ingressClassName of generic ingress
controller.
#  #In case of haproxy the default values is haproxy
#  className: "haproxy"

#provide loadbalancer port
# if TLS is enabled in global section, then loadbalancerport will be used as
external port for Generic or Traefik
loadbalancerport: <loadBalancer-port>
```

Edit **app-messaging-bus.yaml** file as follows:

```
kafka-cluster:
  listeners:
    ingress:
      # if TLS is Disabled in global, then ingressSslPort will be used as
external port.
      ingressSslPort: <LoadBalancer_SSL_Port>
      # If using Generic Ingress controller, below given annotations are
mandatory for Message-Bus external access.
      # These annotations are required for haproxy ingress controller in
Message-Bus.
      annotations:
        haproxy.org/ssl-passthrough: "true"
        ingress.kubernetes.io/ssl-passthrough: "true"
```

In external producer or consumer messaging clients (or applications), the following must be done to access the Kafka cluster through Ingress controller.

Copyright © 2023, 2026, Oracle and/or its affiliates.

- The Bootstrap server and advertised broker host names must be configured in DNS at client side.
- Import the TLS certificate and trust stores from the Kafka cluster into client configurations.
- Add required additional properties in Kafka producer or consumer client configuration.

DNS settings in client applications host

The Bootstrap server host name and advertised broker host names must be configured in /etc/hosts file in producer and consumer client applications with the Ingress Controller Node IP or Load Balancer IP Address. Hostnames are pre-configured when deployed with ingress listener type enabled with the following pattern:

```
bootstrap-server:  <kafka-cluster-instance-name>.<kafka-cluster-project-
name>.messaging.bootstrap.uim.org
broker-0:          <kafka-cluster-instance-name>.<kafka-cluster-project-
name>.messaging.broker0.uim.org
broker-1:          <kafka-cluster-instance-name>.kafka-cluster-project-
name>.messaging.broker1.uim.org

For example if a instance is quick and namesapce is sr then the hostnames
will be as follows:
bootstrap-server:  quick.sr.messaging.bootstrap.uim.org
broker-0:          quick.sr.messaging.broker0.uim.org
broker-1:          quick.sr.messaging.broker1.uim.org
```

> **Note**
>
> You can override the value of **subDomainNameSeparator**. The default value is **"."**, This value can be changed to **"-"** to match the wild card pattern of SSL certificates.
>
> To override the value, uncomment and change it in **applications-base.yaml** as follows:
>
> ```
> #subDomainNameSeparator: "."
>
> #Example hostnames for "-" : quick-sr-messaging-bootstrap.uim.org
> ```

Importing certificates into client applications

See the "Import/export of TLS certificates" section in “Client Access” section for exporting the ca-certs of Kafka cluster to producer or consumer applications.

Message Bus NodePort Listener

This is another external access method which is used by events producer or consumer client applications when they are deployed outside of the Kubernetes cluster and wants to access the message-bus service without ingress controller.

Node port

Copyright © 2023, 2026, Oracle and/or its affiliates.

The following configuration in the application yaml file allows exposing the nodeport listener type to access the Message Bus externally with tls and OAuth 2.0 Authentication.

```
Kafka-cluster:
  listeners:
    #To expose the kafka-cluster to external kafka clients without ingress
controller, uncomment the following section and modify accordingly.
    nodeport:
      tls: true
      # if need to expose on a static nodeport, please uncomment the below
nodePort key and provide values.
      nodePort: 32100
      authentication: true
```

When the TLS is enabled the certificates of the Kafka cluster must be imported in the events producer and consumer clients to access the Kafka cluster.

See the "Import/export of TLS certificates" section in “Client Access” section for exporting the auto-generated ca-certs of Kafka cluster.

Message Bus KRaft Migration

From Message Bus 1.3.0.0.0 onwards, Kafka Message Bus is modified to uptake Zookeeper-less-Kafka (*KRaft*) feature for the metadata managed, which is supported by Apache Kafka. The Kafka nodes in controller role manages this metadata management instead of Zookeeper and these are called Kafka Controllers.

Apache Kafka Raft (KRaft)

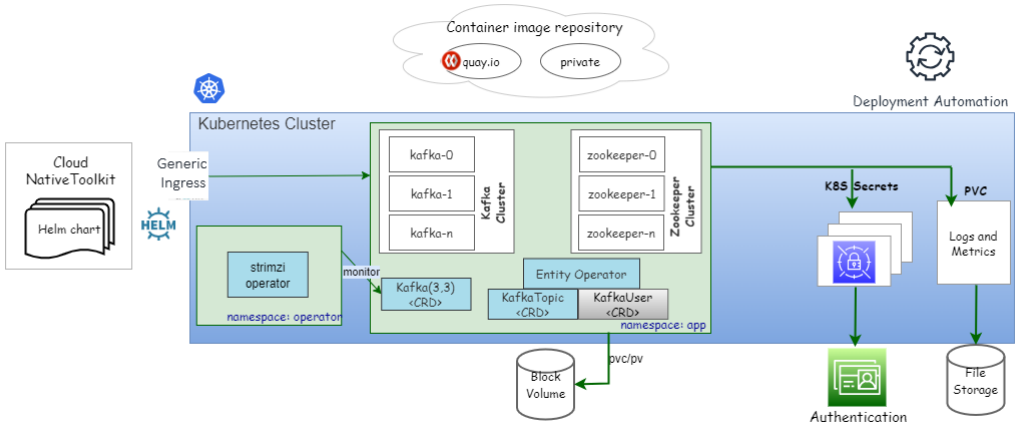
Apache Kafka Raft (KRaft) is the consensus protocol that is introduced in KIP-500 to remove the dependency of Apache Kafka on ZooKeeper for metadata management. KRaft metadata mode replaces Kafka’s dependency on ZooKeeper for cluster management. KRaft mode simplifies the deployment and management of Kafka clusters by bringing metadata management and coordination of clusters into Kafka. Kafka in KRaft mode is designed to offer enhanced reliability, scalability, and throughput. Metadata operations become more efficient as they are directly integrated. Rremoving the need to maintain a ZooKeeper cluster reduces the operational and security overhead.

This migration is a **one-time** activity that migrates Kafka to the KRaft mode. This upgrade and migration can be done without downtime. Once the migration is done, you can use the upgrade application script for any upgrades.

> **Note**
>
> Migrating an ephemeral storage based Kafka Message Bus is not supported for migration and upgrades.

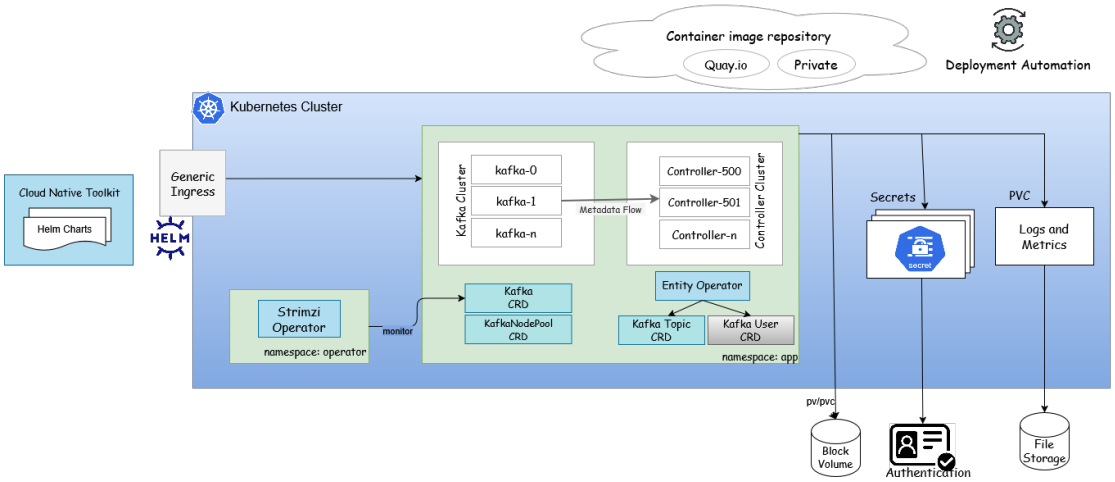
Copyright © 2023, 2026, Oracle and/or its affiliates.

Figure 7-1 Message Bus Architecture with Zookeeper (before Message Bus 1.3.0.0.0)



Once Kafka Message Bus is migrated to KRaft mode, the Zookeeper cluster in the above deployment architecture is converted to have Kafka Controller nodes.

Figure 7-2 Message Bus Architecture (KRaft Mode)



Kafka Message Bus in KRaft mode uses Kafka and KafkaNodePool custom resources to manage Kafka with brokers and controller nodes. The Kafka nodes are assigned with the roles of broker and controller, while controller role replaces Zookeeper):

- **Controller**: These nodes operate in the control plane to manage cluster metadata and the state of the cluster using a Raft-based consensus protocol.
- **Broker**: These nodes operate in the data plane to manage the streaming of messages and receive and store data in topic partitions.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Migrating Message Bus to KRaft Mode

> **Note**
>
> You can perform the Migration of Kafka Message Bus to KRaft mode in either of the following ways:
>
> - If there are in-flight messages in the Kafka topics for consumption, you should run a migration script for migrating the Kafka to KRaft mode.
> - If all the messages from the Kafka topics are consumed, you should delete and create Message Bus service again. If Message Bus uses the default self-generated certificates, you should re-export Message Bus certificate back into the clients (producer or consumer).

This section describes about the process of upgrading the existing Kafka Message Bus to latest version of Kafka where Zookeeper dependency can be removed). To use the other process, see *Unified Inventory and Topology Deployment Guide* to un-install and install the service.

Migration Phases and Configuration Guidance

The conditions for migration are as follows:

- **Before Migration**: The Zookeeper configuration must remain in place and active.
- **During Migration**: Ensure the Zookeeper configuration is present to facilitate a smooth transition.
- **After Migration**: Once the migration is complete, you can remove the Zookeeper configuration from **app-messaging-bus.yaml**, as it will no longer be used.

Prerequisites

The prerequisites for migration are:

- You must have Strimzi-0.45.0 operator or a later version.
- The target Kafka Message Bus namespace (on which migration is being performed) should be registered with the above Strimzi Operator. If the namespace is registered with older Strimzi operator, un-register and register again with the above operator.
- Throughout the migration process, Zookeeper and controller nodes operate in parallel for a period, requiring sufficient compute resources in the cluster.
- Kafka Message Bus should be backed by persistent storage (not to be on ephemeral) for upgrade.
- Oracle recommends you to have multiple Kafka replicas (minimum 3) so that the service is not down while upgrading.

For convenience, a one-item *message-bus-migrate-to-kraft.sh* script is provided for migration. This script will run a sequence of steps to migrate Kafka Cluster to Zookeeper-less mode and in this process it restarts many times.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Run the following script once the prerequisites are met:

```
$COMMON_CNTK/scripts/message-bus-migrate-to-kraft.sh -p <MessageBus
Namespace> -i <MessageBus Instance Name> \
-s <path to spec directory>
```

For example: In the following script, the "sr" is the namespace, and "quick" is used as the instance name:

```
$COMMON_CNTK/scripts/message-bus-migrate-to-kraft.sh -p sr -i quick -
s $SPEC_PATH
```

After the migration starts, you will see that Kafka Message Bus is restarted many times. While the migration is going on, you can see the Kafka cluster changing the state of metadata during from Zookeeper to KRaftPostMigration and from KRaftPostMigration to KRaft state.

Sample Console Output of Kafka Cluster

The sample is as follows:

```
kubectl get kafka sr-quick-messaging -n sr -w

NAME         DESIRED KAFKA REPLICAS   DESIRED ZK REPLICAS
READY          METADATA STATE    WARNINGS
sr-quick-messaging   3                        3
True           ZooKeeper
sr-quick-messaging   3                        3
True           ZooKeeper           True
sr-quick-messaging   3
3                                  ZooKeeper            True
sr-quick-messaging   3                        3
True           KRaftMigration
sr-quick-messaging   3                        3
True           KRaftDualWriting
sr-quick-messaging   3                        3
True           KRaftPostMigration
sr-quick-messaging   3                        3
True           KRaftPostMigration   True
sr-quick-messaging   3                        3
True           KRaftPostMigration   True
sr-quick-messaging   3                        3
True           PreKRaft             True
sr-quick-messaging   3                        3
True           KRaft                True
sr-quick-messaging
True           KRaft                True
sr-quick-messaging
True            KRaft
```

When the migration is successful, the Zookeeper pods will be terminated, and new controller pods will be created as follows:

```
bash-4.4$ kubectl get pods -n test
NAME                                                READY   STATUS
```

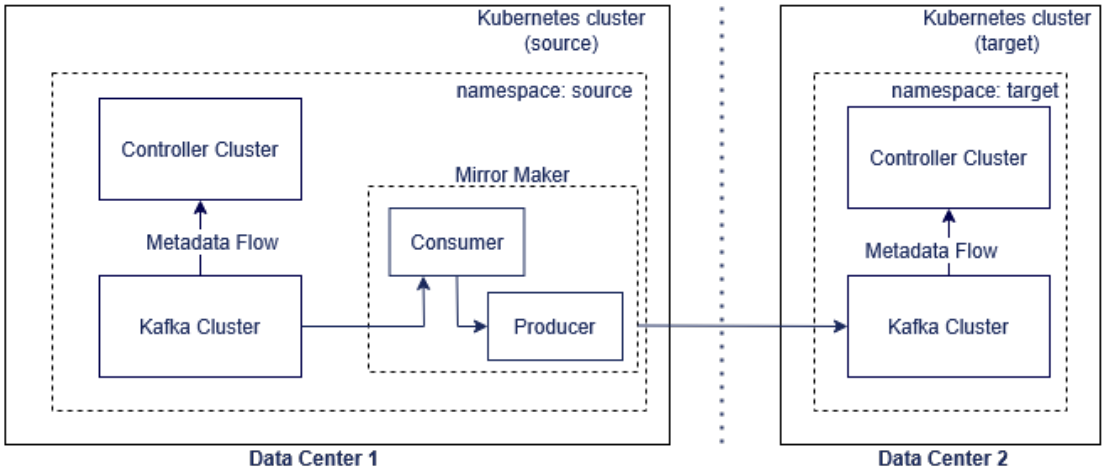

Copyright © 2023, 2026, Oracle and/or its affiliates.

```
RESTARTS   AGE
sr-quick-messaging-controller-500                   1/1     Running
1          6d19h
sr-quick-messaging-controller-501                   1/1     Running
1          6d19h
sr-quick-messaging-controller-502                   1/1     Running
1          6d19h
sr-quick-messaging-entity-operator-878f95b96-8jddm  1/1     Running
1          6d19h
sr-quick-messaging-kafka-0                          1/1     Running
1          6d19h
sr-quick-messaging-kafka-1                          1/1     Running
1          6d19h
sr-quick-messaging-kafka-2                          1/1     Running
1          6d19h
```

Geo Redundancy Support

The Geo Redundancy of Message Bus (which uses Kafka) is achieved with Mirror Maker tool. Apache Kafka Mirror Maker replicates data across two Kafka clusters, within or across data centers. See https://strimzi.io/blog/2020/03/30/introducing-mirrormaker2/ for more details.

The following diagram shows an example of how mirror maker replicates the topics from source Kafka cluster to target Kafka cluster.



The prerequisites are as follows:

- The Strimzi operator should be up and running
- The source Message Bus service should be up and running
- The target Message Bus service should be up and running

Strimzi Operator

Validate that the Strimzi operator is installed by running the following command:

```
$kubectl get pod -n <STRIMZI_NAMESPACE>
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
NAME                                          READY   STATUS    RESTARTS   AGE
strimzi-cluster-operator-566948f58c-sfj7c   1/1          Running
0          6m55s
```

Validate installed helm release for Strimzi operator by running the following command:

```
$helm list -n <STRIMZI_NAMESPACE>

NAME                    NAMESPACE          REVISION        STATUS
CHART                         APP VERSION
strimzi-operator        STRIMZI_NAMESPACE  1               deployed
strimzi-kafka-operator-0.X.0   0.X.0
```

Source Message Bus

The source Message Bus should be up and running (the Kafka cluster from which the topics should be replicated).

Validate the Kafka cluster is installed by running the following command:

```
$kubectl get pod -n sr1

 NAME                                                    READY   STATUS
RESTARTS   AGE
 sr1-quick1-messaging-entity-operator-5f9c688c7-2jcjg      1/1
Running    0          27h
 sr1-quick1-messaging-kafka-0                            1/1
Running    0          27h
 sr1-quick1-messaging-controller-500                     1/1
Running    0          27h
```

Validate the persistent volume claims created for the Kafka cluster by running the following command:

```
$kubectl get pvc -n sr1

NAME                                       STATUS
VOLUME                                    CAPACITY   ACCESS MODES
STORAGECLASS   AGE
data-sr1-quick1-messaging-kafka-0          Bound    pvc-0a153e05-
df71-431e-9fca-d2bb5b55c701  1Gi        RWO           sc             27h
data-sr1-quick1-messaging-controller-500   Bound    pvc-888e1926-5f5e-4541-
b3f7-2c4647b7cb6b  1Gi        RWO           sc
```

Target Message Bus

The target Message Bus should be up and running (the Kafka cluster to which the topics should be replicated).

Validate the Kafka cluster is installed by running the following command:

```
$kubectl get pod -n sr2

 NAME                                                    READY   STATUS
RESTARTS   AGE
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
 sr2-quick2-messaging-entity-operator-5f9c688c7-2jcjg     1/1
Running   0          27h
 sr2-quick2-messaging-kafka-0                             1/1
Running   0          27h
 sr2-quick2-messaging-controller-500                      1/1
Running   0          27h
```

Validate the persistent volume claims created for the Kafka cluster by running the following command:

```
$kubectl get pvc -n <kafka target namespace>`

NAME                                        STATUS
VOLUME                                     CAPACITY   ACCESS MODES
STORAGECLASS   AGE
data-sr2-quick2-messaging-kafka-0          Bound    pvc-0a153e05-
df71-431e-9fca-d2bb5b55c701  1Gi        RWO           sc             27h
data-sr2-quick2-messaging-controller-500   Bound    pvc-888e1926-5f5e-4541-
b3f7-2c4647b7cb6b  1Gi        RWO           sc             27h
```

Installing and Configuring Mirror Maker 2.0

This section describes the installation and configuration of Mirror Maker 2.0.

Configuring Source and Target Message Bus (Kafka cluster) Details

Update the *$COMMON_CNTK/samples/messaging-bus/kafka-mirror-maker/values.yaml* with source and target Kafka cluster details as follows:

```
sourceCluster:
    #Source Kafka cluster
    name: sr1-quick1-messaging
    #Bootstarp server for connection to the source Kafka cluster
    bootstrapServers: sr1-quick1-messaging-kafka-bootstrap:9092
  targetCluster:
    #Target Kafka cluster
    name: sr2-quick2-messaging
    #Bootstarp server for connection to the target Kafka cluster
    bootstrapServers: sr2-quick2-messaging-kafka-bootstrap:9092
```

In the above command:

- **sourceCluster.name** is the helm release for source Kafka cluster (sr1-quick1-messaging)
- **sourceCluster.bootstrapServers** is the bootstrap server of source kafka cluster (**sr1-quick1-messaging-kafka-bootstrap.test.svc:9092**). When the source cluster is using Ingress as an external listener, **sourceCluster.bootstrapServers** is **quick1.sr1.messaging.bootstrap.uim.org:<loadbalancerPort>**.
- **targetCluster.name** is the helm release for target Kafka cluster (sr2-quick2-messaging)
- **targetCluster.bootstrapServers** is the bootstrap server of target Kafka cluster (**sr2-quick2-messaging-kafka-bootstrap:9092**). When target cluster is using Ingress as an external listener, **targetCluster.bootstrapServers** is **quick2.sr2.messaging.bootstrap.uim.org:<loadbalancerPort>**.

Copyright © 2023, 2026, Oracle and/or its affiliates.

> **Note**
>
> To enable geo replication between the Kafka clusters from different namespaces, we can use the hostname pattern as **servicename.namespace.svc.cluster.local** while updating
>
> `$COMMON_CNTK/samples/messaging/kafka-mirror-maker/values.yaml`
>
> If the **sr1-quick1-messaging-kafka-bootstrap** service is hosted in Strimzi namespace on 9092 port and the client application in another namespace, then the bootstrap-server URL should be used as **sr1-quick1-messaging-kafka-bootstrap.strimzi.svc.cluster.local**
>
> If the target cluster is in another Kubernetes cluster, you must to use external listener for referring to the boostrap server.
>
> While using `Nodeport`, the worker node IP of the target cluster is to be used as the target cluster bootstrap address along with the exposed nodeport.
>
> While using `Ingress`, the hostname of the target cluster is to be used as target cluster bootstrap address.

Configuring Ingress for Source and Target Cluster

If Ingress is enabled for either the source or target cluster, add **hostAliases** to resolve the Ingress hostname to the correct IP address inside the pod. The number of hostnames depends on the Kafka replica count in the source or target cluster. If the Kafka replica size is **3**, there will be 3 broker DNS entries and 1 bootstrap DNS entry.

And when the Ingress is enabled in a cluster, the TLS is enabled by default and required. To allow mirror maker to securely access the source or target clusters, the Kafka cluster ingress listener configured certificates must be added to the mirror maker's **tlsTrustedCertificates** section. This ensures that mirror maker can trust the TLS certificates used by the Kafka clusters.

Sample configuration of Source Cluster:

```
sourceCluster:
  listeners:
    ingress:
      hostAliases:
        - ip: "Ip-Address"
          hostnames:
            - "quick1.sr1.messaging.bootstrap.uim.org"
            - "quick1.sr1.messaging.broker0.uim.org"
            - "quick1.sr1.messaging.broker1.uim.org"
            - "quick1.sr1.messaging.broker2.uim.org"
      tlsTrustedCertificates:
        - secretName: sr1-quick1-messaging-cluster-ca-cert
          certificate: ca.crt
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

Sample configuration for Target Cluster:

```
targetCluster:
  listeners:
    ingress:
      hostAliases:
        - ip: "Ip-Address"
          hostnames:
            - "quick2.sr2.messaging.bootstrap.uim.org"
            - "quick2.sr2.messaging.broker0.uim.org"
            - "quick2.sr2.messaging.broker1.uim.org"
            - "quick2.sr2.messaging.broker2.uim.org"
      tlsTrustedCertificates:
        - secretName: sr2-quick2-messaging-cluster-ca-cert
          certificate: ca.crt
```

Exporting certificate:

Use the `$COMMON_CNTK/scripts/export-message-bus-cert.sh -p sr1 -i quick1 -l . -k ./mb-test-client-cert-keystore-kafka.jks -a mb-cert` script to extract Kafka cluster certificate.

This generates a certificate **sr1-quick1-messaging.crt**.

Then use the follow command to create the secret in mirror maker namespace. Do this for both Source and Target Clusters:

```
kubectl create secret generic sr1-quick1-messaging-cluster-ca-cert  --from-
file=ca.crt=sr1-quick1-messaging.crt   -n mirror-maker
```

Configuring OAuth for Source and Target Cluster

If the Source and Target cluster has OAuth enabled, provide the following values:

```
sourceCluster:
    name: sno-dev-messaging  #source kafka cluster #Helm Release Name
    # Authentication to Connect to source Cluster
    bootstrapServers: sno-dev-messaging-kafka-bootstrap.test.svc:9092
#Bootstarp server for connection to the source Kafka cluster
    authentication:
      enabled: true
      type: oauth
      clientId: <Client ID>
      clientSecret:
        key: <Key>
        secretName: <Secret Name>
      tokenEndpointUri: <Token Endpoint URi>
       #tlsTrustedCertificates are required only if target cluster has tls
enabled on their oauth server
      tls: true
       #Uncomment the below if tls is enabled in oauth
      tlsTrustedCertificates:
      - secretName:  <Secret Name>
        certificate: idpcert.pem
  targetCluster:
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
    name: sno2-dev-messaging  #target kafka cluster #Helm Release Name
    bootstrapServers: sno2-dev-messaging-kafka-bootstrap.test.svc:9092
#Bootstarp server for connection to the target Kafka cluster
    # Authentication to connect to target cluster
    authentication:
      enabled: true
      type: oauth
      clientId: <Client ID>
      clientSecret:
        key: <Key>
        secretName: <Secret Name>
      tokenEndpointUri: <Token Endpoint URi>
       #tlsTrustedCertificates are required only if target cluster has tls
enabled on their oauth server
      tls: true
       #Uncomment the below if tls is enabled in oauth
      tlsTrustedCertificates:
      - secretName:  <Secret Name>
        certificate: idpcert.pem
```

If TLS is enabled in the identity provider, all OAuth related details must be provided in **values.yaml**. Additionally, a secret containing the OAuth TLS certificate and Client Secret must be created in the namespace where mirror maker is deployed.

For example:

```
kubectl create secret generic mirror-maker-Oauth-Source   --from-
file=idpcert.pem=keycloak.pem   --from-literal=oauth_client_secret=<client-
secret>   -n mirror-maker

kubectl create secret generic mirror-maker-Oauth-target   --from-
file=idpcert.pem=keycloak.pem   --from-literal=oauth_client_secret=<client-
secret>   -n mirror-maker
```

Configuring Metrics for Mirror Maker

To enhance monitoring capabilities of a mirror maker cluster using JVM metrics, you can enable the JMX Exporter in your MirrorMaker2 configuration using the following commands:

```
metrics:
  jmxExporter:
    enable: false
```

When enabled, the JMX Exporter provides valuable insights into your MirrorMaker cluster's performance. It is a crucial component for:

- **Replication Health Monitoring**: Ensuring data replication is functioning as expected.
- **Lag Measurement**: Measuring the delay between the source and target Kafka clusters.
- **Thread Management**: Monitoring MirrorMaker threads to avoid overutilization.

These metrics are the foundation of observability in complex Kafka setups, especially in hybrid or multi-cluster architectures. You can visualize these metrics using:

Copyright © 2023, 2026, Oracle and/or its affiliates.

- **Prometheus UI**: By configuring a Scrape job, you can view the metrics on the Prometheus interface.
- **Grafana Dashboard**: This tool provides graphical representations of the metrics, making it easier to monitor.

See "Managing Message Bus Metrics" for more information.

Installing Mirror Maker

Run the following command to install Mirror Maker in specific namespace:

```
helm install mirror-maker $COMMON_CNTK/samples/messaging/kafka-mirror-maker/ -
n <namespace> --values $COMMON_CNTK/samples/messaging/kafka-mirror-maker/
values.yaml
```

Validate that Mirror Maker is installed by running the following command:

```
kubectl get pods -n <namespace>
replication-mirror-maker-mirrormaker2-5c6d7dd7d7-r89cj             1/1
Running   0          67m
kubectl get svc -n <namespace>
replication-mirror-maker-mirrormaker2-api          ClusterIP   <clusterIP>
<none>        8083/TCP                              67m
```

Uninstalling Mirror Maker

Run the following command to uninstall Mirror Maker from specific namespace:

```
helm uninstall mirror-maker -n <namespace>
```

Delete topic mm2-offset-syncs.messaging-test.internal from the source cluster (dev1-messaging)

```
$kubectl -n <SourceKafkaClusterNamespace> run kafka-topic -ti --
image=<STRIMZI_KAFKA_IMAGE_NAME> --rm=true --restart=Never -- bin/kafka-
topics.sh --bootstrap-server <instance>-messaging-kafka-bootstrap:9092 --
delete --topic mm2-offset-syncs.messaging-test.internal
```

Delete topics heartbeats, mirrormaker2-cluster-status, mirrormaker2-cluster-offsets, mirrormaker2-cluster-configs from the target cluster (dev2-messaging)

```
$kubectl -n <TargetKafkaClusterNamespace> run kafka-topic -ti --
image=<STRIMZI_KAFKA_IMAGE_NAME> --rm=true --restart=Never -- bin/kafka-
topics.sh --bootstrap-server <namespace>-<instance>-messaging-kafka-
bootstrap:9092 --delete --topic heartbeats

$kubectl -n <TargetKafkaClusterNamespace> run kafka-topic -ti --
image=<STRIMZI_KAFKA_IMAGE_NAME> --rm=true --restart=Never -- bin/kafka-
topics.sh --bootstrap-server <namespace>-<instance>-messaging-kafka-
bootstrap:9092 --delete --topic mirrormaker2-cluster-status

$kubectl -n <TargetKafkaClusterNamespace> run kafka-topic -ti --
image=<STRIMZI_KAFKA_IMAGE_NAME> --rm=true --restart=Never -- bin/kafka-
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
topics.sh --bootstrap-server <namespace>-<instance>-messaging-kafka-
bootstrap:9092 --delete --topic mirrormaker2-cluster-offsets

$kubectl -n <TargetKafkaClusterNamespace> run kafka-topic -ti --
image=<STRIMZI_KAFKA_IMAGE_NAME> --rm=true --restar
```

Debugging and Troubleshooting

NotEnoughReplicasException

When you get the **org.apache.kafka.common.errors.NotEnoughReplicasException**: Messages are rejected since there are fewer in-sync replicas than required. The reason could be that the topics replicas is not meeting the default minInsyncReplicas value configured in the Message Bus service.

Asynchronous auto-commit of offsets failed

Problem

When you get the following error in the logs (for example: ATA Consumer). .

```
[Consumer clientId=consumer-ora-uim-topology-service-2, groupId=ora-uim-
topology-service] Asynchronous auto-commit of offsets failed: Offset commit
cannot be completed since the consumer is not part of an active group for
auto partition assignment; it is likely that the consumer was kicked out of
the group.. Will continue to join group.
```

Resolution

To resolve this make sure that **max.polling.interval.ms** is always greater than the last poll or else reduce the **max.poll.records**.

Add these additional properties in the YAML file under the **mp.messaging.connector.helidon-kafka** section with override values.

```
mp.messaging:
  connector:
    helidon-kafka:
      # The following are default global configuration values which effects
for all the consumer groups.
      max.polling.interval.ms: 300000
      max.poll.records: 500

  # The following are channel specific configuration values
  incoming:
    # The toInventoryChannel effects only for ora-uim-topology-service
consumer group
    # uncomment and update the specific values
    #toInventoryChannel:
      #max.polling.interval.ms: 300000
      #max.poll.records: 500

    # The toFaultChannel effects only for ora-uim-topology-retry-service
consumer group
    # Uncomment and update the specific values
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
    #toRetryChannel:
      #max.polling.interval.ms: 300000
      #max.poll.records: 200

    # The toDltChannel effects only for ora-uim-topology-dlt-service consumer
group
    # uncomment and update the specific values
    #toDltChannel:
      #max.polling.interval.ms: 300000
      #max.poll.records: 100
```

Performance Tuning: Consumer Configurations

The following are some consumer configuration properties in message consumers which are related to performance. See https://kafka.apache.org/documentation/#consumerconfigs for all available consumer config properties.

- max.poll.records (default=500) defines the maximum number of messages that a consumer can poll at once.
- max.partition.fetch.bytes (default=1048576) defines the maximum number of bytes that the server returns in a poll for a single partition.
- max.poll.interval.ms (default=300000) defines the time a consumer must process all messages from a poll and fetch a new poll afterward. If this interval is exceeded, the consumer leaves the consumer group.
- http://heartbeat.interval.ms (default=3000) defines the frequency with which a consumer sends heartbeats.
- http://session.timeout.ms (default=10000) defines the time a consumer must send a heartbeat. If no heartbeat was received in that timeout, the member is considered dead and leaves the group.

Managing Consumer Groups

For more list of options available on the consumer groups see the apache kafka managing consumer groups section. The following sub-sections list some significant operations.

See "Message Bus Client Access" on how to run the message bus test client pod with required configuration.

List consumer groups

```
#Exec into running message bus test client pod
kubectl exec -it mb-test-auth-client-deployment-******-**** -n sr -- bash

#Run the below command to list all the consumer groups
bin/kafka-consumer-groups.sh \
--command-config /home/kafka/mb_test_client.properties \
--bootstrap-server <Your Bootstrap Server URL> \
--list
```

Describe consumer group

```
#Exec into running Kafka admin client pod
kubectl exec -it mb-test-auth-client-deployment-******-**** -n sr -- bash
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
#Run the below command to describe specific consumer group to check topics,
partitions, offsets
#Replace the command-config, bootstrap, group values accordingly
bin/kafka-consumer-groups.sh \
--command-config /home/kafka/mb_test_client.properties \
--bootstrap-server <Your Bootstrap Server URL> \
--group test-client-service \
--describe
```

Reset offset of a consumer group

```
#Exec into running Kafka admin client pod
kubectl exec -it mb-test-auth-client-deployment-******-**** -n sr -- bash

#Run the below command to reset offset for consumer group for topic to
latest.  See Apache Kafka documentation for other available options.
#Replace the command-config, bootstrap, group and topic values accordingly
bin/kafka-consumer-groups.sh \
--command-config /home/kafka/mb_test_client.properties \
--bootstrap-server <Your Bootstrap Server URL> \
--group test-client-service \
--reset-offsets \
--topic ora-test-topic \
--to-latest \
--execute
```

Topics

For more detailed list of operations available on the topics see the "Apache Kafka Operations". The following sub-sections list some significant operations.

See "Message Bus Client Access" section for more information.

Create

Create a topic with three partitions and two replications.

```
#Exec into running Kafka admin client pod
kubectl exec -it mb-test-auth-client-deployment-******-**** -n sr -- bash


#Run the below command to create a topic
bin/kafka-topics.sh \
 --command-config /home/kafka/mb_test_client.properties \
 --bootstrap-server <Your Bootstrap Server URL> \
 --create \
 --topic replicated-2 \
 --replication-factor 2 \
 --partitions 3
```

List

To list all topics:

```
#Exec into running Kafka admin client pod
kubectl exec -it mb-test-auth-client-deployment-******-**** -n sr -- bash
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
#Run the below command to list all the topic
bin/kafka-topics.sh \
 --command-config /home/kafka/mb_test_client.properties \
 --bootstrap-server <Your Bootstrap Server URL> \
 --list
```

Describe

Describes the topic and its partition count, replicas factory along with leaders for the partition.

```
#Exec into running Kafka admin client pod
kubectl exec -it mb-test-auth-client-deployment-******-**** -n sr -- bash

#Run the below command to describe the topic
bin/kafka-topics.sh \
 --command-config /home/kafka/mb_test_client.properties \
 --bootstrap-server <Your Bootstrap Server URL> \
 --topic replicated-2 \
 --describe

#Sample output
Topic: replicated-2      TopicId: vyalpPOmR0CtYt7Sc-gbxA
PartitionCount: 3               ReplicationFactor: 2          Configs:
min.insync.replicas=1,message.format.version=3.0-IV1

Topic: replicated-2     Partition: 0     Leader: 1      Replicas: 1,0
Isr: 1,0
Topic: replicated-2     Partition: 1     Leader: 0      Replicas: 0,1
Isr: 0,1
Topic: replicated-2     Partition: 2     Leader: 1      Replicas: 1,0
Isr: 1,0
```

Alter

You can alter a topic and increase the partitions to 2.

```
#Exec into running Kafka admin client pod
kubectl exec -it mb-test-auth-client-deployment-******-**** -n sr -- bash

#Run the below command to alter the topic  bin/kafka-topics.sh \
 --command-config /home/kafka/mb_test_client.properties \
 --bootstrap-server <Your Bootstrap Server URL> \
 --alter \
 --topic <Your Topic Name> \
 --partitions 1
```

Reassignment

The partition reassignment tool can also be used to selectively move replicas of a partition to a specific set of brokers. In the following example the partitions for topic (**replicated-2**) are reassigned to different brokers.

See the "Message Bus Client Access" section for more information on running the message bus test pod with required configuration such as Authentication and SSL.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Create a file called **custom-reassignment.json** file a terminal

```
{"version":"1", "partitions":
[{"topic":"replicated-2","partition":"0","replicas":"[0,1]"},
{"topic":"replicated-2","partition":1,"replicas":"[1,2]"},
{"topic":"replicated-2","partition":"2","replicas":"[0,2]"}]}
```

Run the following commands for reassignment:

```
#Copy the custom-reassignment.json file into the newly created pod under /
home/kafka directory
$kubectl cp custom-reassignment.json mb-test-auth-client-deployment-*****-
****:/home/kafka/custom-reassignment.json -n kafka


#Exec into running test pod
kubectl exec -it mb-test-auth-client-deployment-******-**** -n sr -- bash
#Cd directory to /home/kafka

#Validate the topic ("replicated-2"
/opt/kafka/bin/kafka-topics.sh \
--command-config /home/kafka/mb_test_client.properties \
--bootstrap-server <Your Bootstrap Server URL> \
--topic replicated-2 --describe
Topic: replicated-2     TopicId: vyalpPOmR0CtYt7Sc-gbxA PartitionCount:
3       ReplicationFactor: 2    Configs:
min.insync.replicas=1,message.format.version=3.0-IV1
        Topic: replicated-2     Partition: 0    Leader: 1       Replicas:
1,0   Isr: 1,0
        Topic: replicated-2     Partition: 1    Leader: 1       Replicas:
0,1   Isr: 1,0
        Topic: replicated-2     Partition: 2    Leader: 1       Replicas:
1,0   Isr: 1,0

#Run reassign-partitions script to reassign the partitions according to the
json file
$/opt/kafka/bin/kafka-reassign-partitions.sh --bootstrap-server dev-messaging-
kafka-bootstrap:9092 --reassignment-json-file custom-reassignment.json --
execute

Current partition replica assignment

{"version":1,"partitions":[{"topic":"replicated-2","partition":0,"replicas":
[1,0],"log_dirs":["any","any"]},
{"topic":"replicated-2","partition":1,"replicas":[0,1],"log_dirs":
["any","any"]},{"topic":"replicated-2","partition":2,"replicas":
[1,0],"log_dirs":["any","any"]}]}

Save this to use as the --reassignment-json-file option during rollback
Successfully started partition reassignments for
replicated-2-0,replicated-2-1,replicated-2-2

#Verfify the reassignment status
$/opt/kafka/bin/kafka-reassign-partitions.sh --bootstrap-server dev-messaging-
kafka-bootstrap:9092 --reassignment-json-file custom-reassignment.json --
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
verify

Status of partition reassignment:
Reassignment of partition replicated-2-0 is complete.
Reassignment of partition replicated-2-1 is complete.
Reassignment of partition replicated-2-2 is complete.

Clearing broker-level throttles on brokers 0,1,2
Clearing topic-level throttles on topic replicated-2

# Validate the partition assignments
$/opt/kafka/bin//kafka-topics.sh \
  --command-config /home/kafka/mb_test_client.properties \
  --bootstrap-server <Your Bootstrap Server URL> \
  --topic replicated-2 --describe
Topic: replicated-2     TopicId: vyalpPOmR0CtYt7Sc-gbxA PartitionCount:
3       ReplicationFactor: 2    Configs:
min.insync.replicas=1,message.format.version=3.0-IV1
        Topic: replicated-2     Partition: 0    Leader: 1       Replicas:
0,1   Isr: 1,0
        Topic: replicated-2     Partition: 1    Leader: 1       Replicas:
1,2   Isr: 1,2
        Topic: replicated-2     Partition: 2    Leader: 0       Replicas:
0,2   Isr: 0,2
```

See, , https://cwiki.apache.org/confluence/display/KAFKA/Replication+tools#Replicationtools-4.ReassignPartitionsTool for more information.

ValidationException: Failed to Extract Principal

Problem

The exception you receive is as follows:

```
Failed to extract principal - check usernameClaim, fallbackUsernameClaim
configuration
```

When you see the following exception in Message Bus with `DEBUG logs` enabled, the `sub` could be missing in the validate token end-point URL of IDP.

```
io.strimzi.kafka.oauth.validator.ValidationException: Failed to extract
principal - check usernameClaim, fallbackUsernameClaim configuration
        at
io.strimzi.kafka.oauth.validator.OAuthIntrospectionValidator.validate(OAuthInt
rospectionValidator.java:348)
        at
io.strimzi.kafka.oauth.server.JaasServerOauthValidatorCallbackHandler.validate
Token(JaasServerOauthValidatorCallbackHandler.java:668)
```

Resolution:

You can resolve this exception in the following ways:

- Update the validate token end-point of IDP to return the `sub` attribute.

Copyright © 2023, 2026, Oracle and/or its affiliates.

- Update the authentication additional optional configuration section of Message Bus in **app-messaging-bus.yaml** to pass the replacement field for the `userNameClaim`.

See "Configuring Authentication for Services" for more information.

Sample Configuration

Add the following to **app-messaging-bus.yaml** file if the replacement field is client_id:

```
listeners:
  authentication:
    oauthConfig:
      userNameClaim: client_id
```

Cluster Operator does not start on Kubernetes 1.33

Problem

On Kubernetes v1.33, the Strimzi Cluster Operator (pre-0.45.1) fails to start due to an inability to decode the version string returned by the Fabric8 Kubernetes Client

See https://github.com/fabric8io/kubernetes-client/issues/7037 for more information.

Temporary Fix

If you're using Strimzi ≤ 0.46.0, apply this workaround:

```
kubectl set env deployment/strimzi-cluster-operator
STRIMZI_KUBERNETES_VERSION="major=1,minor=33"
```

For more information, see https://github.com/strimzi/strimzi-kafka-operator/issues/11386

See "Configuring Authentication for Services" for more information.

Sample Configuration

Add the following to **application.yaml** file if the replacement field is client_id:

```
listeners:
  authentication:
    oauthConfig:
      userNameClaim: client_id
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

8

Deploying the Active Topology Automation Service

This chapter describes how to deploy and manage ATA service.

Overview of ATA

Oracle Communications Active Topology Automation (ATA) represents the spatial relationships among your inventory entities for the inventory and network topology.

- ATA provides a graphical representation of topology where you can see your inventory and its relationships at the level of detail that meets your needs.

  See ATA Help for more information about the topology visualization.

Use ATA to view and analyze the network and service data in the form of topology diagrams. ATA collects this data from UIM.

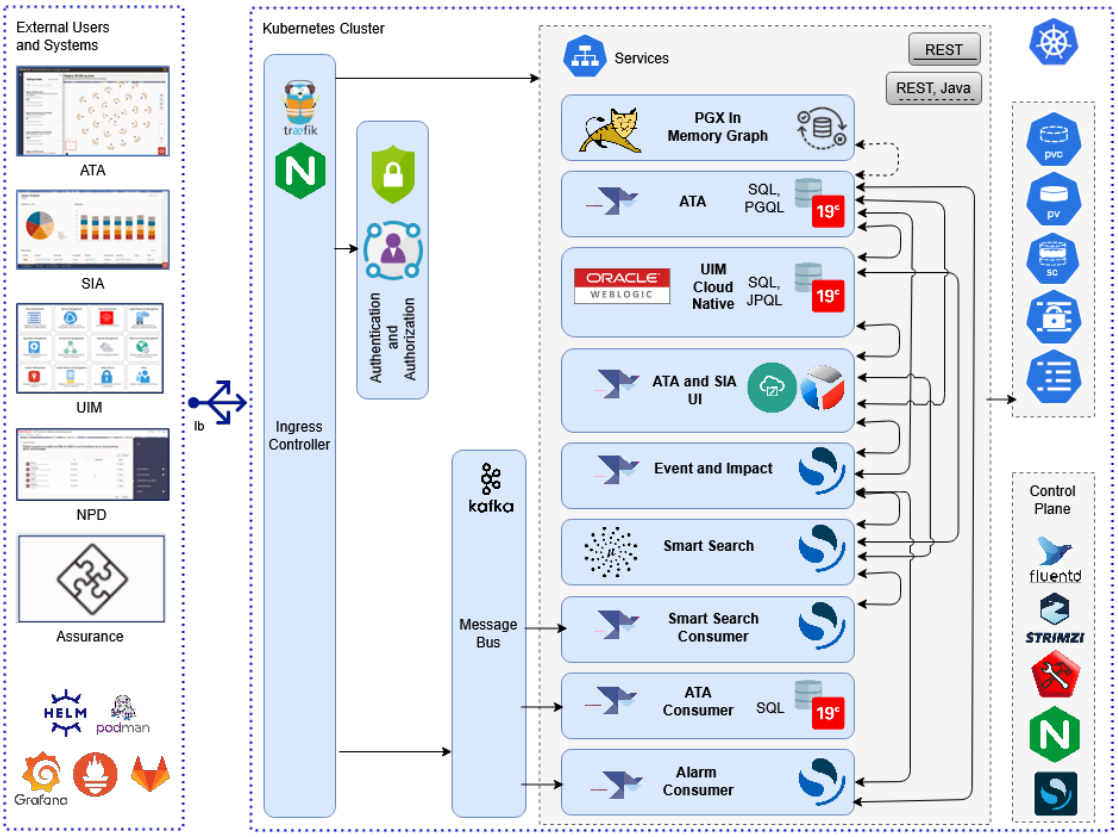
You use ATA for the following:

- Viewing the networks and services, along with the corresponding resources, in the form of topological diagrams and graphical maps.
- Planning the network capacity.
- Tracking networks.
- Viewing alarm information.

ATA Architecture

The following figure shows a high-level architecture of the ATA service.

Copyright © 2023, 2026, Oracle and/or its affiliates.



> **Note**
>
> ATA refers to both Active Topology Automation and Service Impact Analysis.

UIM as the Producer

UIM communicates with the Topology Service using REST APIs and Kafka Message Bus. UIM is the Producer for Create, Update and Delete operations from UIM that impact Topology. UIM uses REST APIs to communicate directly with the ATA Service while building the messages and can also continue processing when the Topology Service is unavailable.

ATA Consumer

The ATA consumer service is a message consumer for topology updates from UIM using Message Bus service. ATA consumer processes the message of TopologyNodeCreate, TopologyNodeUpdate, TopologyNodeDelete, TopologyEdgeCreate, TopologyEdgeUpdate, TopologyEdgeDelete, TopologyProfileDelete, TopologyProfileDelete, and TopologyProfileUpdate event types. See "ATA Events and Definitions" chapter in *Active Topology and Automation Asynchronous Events Guide*.

Alarm Consumer

The alarm consumer service is a Kafka consumer client (implemented as **Helidon MP** microservice) listening on the *ora-alarm-topology* topic for the *fault* and *performance* alarm

Copyright © 2023, 2026, Oracle and/or its affiliates.

event notifications. The alarm event notification should be of TMF642 alarm JSON wrapped in TMF688 event JSON. The alarm consumer supports the TMF alarm event notification v5.0 specification. See https://www.tmforum.org/oda/open-apis/directory/alarm-management-api-TMF642/v5.0 for more information. The supported event types in this release are **AlarmCreateEvent**, **AlarmAttributeValueChangeEvent**, **ClearAlarmCreateEvent**, and **AlarmDeleteEvent**. See the *Active Topology Automation Asynchronous Events Guide* for sample alarm payload and message header information.

Alarm consumer creates the alarms and associate them with the affected entity (node or sub-node) and manage (update, clear, or delete) it for sub-sequent alarm event notifications. For each of the alarm event notification, it creates an event in the Service Impact Analysis service for trouble to resolve the use cases. The alarm consumer finds the affected entity (device or sub-device) in the inventory (UIM or ATA) by filtering the details available from the alarm event notification and associates alarm to it. By default, it refers the **alarmedObject** sub-structure in the alarm event notification and parse the **alarmedObject.id** and **alarmObject.@referredType** fields to find the entity in UIM or ATA. See "*configuring alarm consumer*" for more information on default logic and extension provided to customize the default behavior.

SmartSearch API

SmartSearch API enables you to search, filter, autocompletes, or aggregates the bulk API to batch process for insert, update, or delete top entries from the OpenSearch database.

SmartSearch Consumer

SmartSearch is a consumer for UIM. It processes multiple message events such as TopologyNodeCreate, TopologyNodeUpdate, TopologyNodeDelete, TopologyEdgeCreate, TopologyEdgeUpdate, TopologyEdgeDelete and so on.

OpenSearch

OpenSearch is NoSQL database it is an open-source search and analytics suite that makes it easy to ingest, search, visualize, and analyze data.

Topology Graph Database

The ATA Service communicates to the Oracle Databases using the Oracle Property Graph feature with PGQL and standard SQL. It can communicate directly to the database or with the In-Memory Graph for high performance operations. This converged database feature of Oracle Database makes it possible to utilize the optimal processing method with a single database. The Graph Database is isolated and a separate Pluggable Database (PDB) from the UIM Database but runs on the same 19c version for simplified licensing.

PGX In-Memory Graph

The ATA Service also uses the Oracle Labs Parallel Graph AnalytiX (PGX) In-Memory database. The PGX server is used for Path Analysis and is configured for periodic updates.

ATA User Interface

ATA provides a graphical representation of topology where you can see your inventory and its relationships at the level of detail that meets your needs. ATA is built using Oracle Redwood Design System.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Prerequisites and Configuration for Creating ATA Images

You must install the prerequisite software and tools for creating ATA images.

Prerequisites for Creating ATA Images

You require the following prerequisites for creating ATA images:

- Podman on the build machine if Linux version is greater than or equal to 8.
- Docker on the build machine if Linux version is lesser than 8
- ATA Builder Toolkit (ref about the deliverables)
- Install Maven and update path variable with Maven Home.

  ```
  Set PATH variable export PATH=$PATH:$MAVEN_HOME/bin
  ```

- Java, installed with JAVA_HOME set in the environment.

  ```
  Set PATH variable export PATH=$PATH:$JAVA_HOME/bin
  ```

- Bash, to enable the `<tab>` command complete feature.

See "UIM Software Compatibility" in *UIM Compatibility Matrix* for details about the required and supported versions of these prerequisite software.

Configuring ATA Images

The dependency manifest file describes the input that goes into the ATA images. It is consumed by the image build process. The default configuration in the latest manifest file provides the necessary components for creating the ATA images easily. See "About the Manifest File" for more information.

Creating ATA Images

To create the ATA images:

> **Note**
>
> See "UIM Software Compatibility" in *UIM Compatibility Matrix* for the latest versions of software.

1. Go to WORKSPACEDIR.
2. Download graph server war file from Oracle E-Delivery (https://edelivery.oracle.com/osdc/faces/SoftwareDelivery → Oracle Graph Server <*version*>→ Oracle Graph Webapps <*version*> for (Linux x86-64)) and copy graph server war file to directory $WORKSPACEDIR/ata-builder/staging/downloads/graph. Ensure only one copy of PGX.war exists in …/downloads/graph path.

Copyright © 2023, 2026, Oracle and/or its affiliates.

> **Note**
>
> The log level is set to debug by default in graph server war file. If required, update the log level to **error/info in graph-server-webapp-<*version*>.war/WEB-INF/classes/logback.xml** before building images.

3. Download **apache-tomcat-<*tomcat_version*>.tar.gz** and copy to $WORKSPACEDIR/ata-builder/staging/downloads/tomcat.
4. Download **jdk-<*jdk_version*>_linux-x64_bin.tar.gz** and copy to $WORKSPACEDIR/ata-builder/staging/downloads/java.

> **Note**
>
> For Tomcat and JDK versions, see Unified Inventory and Topology Microservices

5. Export proxies in environment variables, fill the details on proxy settings:

```
#The eth0 is sample. replace "etho" with your specific interface name.
export ip_addr=`ip -f inet addr show eth0|egrep inet|awk '{print $2}'|awk -
F/ '{print $1}'`
export http_proxy=
export https_proxy=$http_proxy
export no_proxy=localhost,$ip_addr
export HTTP_PROXY=
export HTTPS_PROXY=$HTTP_PROXY
export NO_PROXY=localhost,$ip_addr
```

6. Update $WORKSPACEDIR/ata-builder/bin/gradle.properties with required proxies.

```
systemProp.http.proxyHost=
systemProp.http.proxyPort=
systemProp.https.proxyHost=
systemProp.https.proxyPort=
systemProp.http.nonProxyHosts=localhost|127.0.0.1
systemProp.https.nonProxyHosts=localhost|127.0.0.1
```

7. Uncomment the proxy block and provide $WORKSPACEDIR/ata-builder/bin/m2/settings.xml with required proxies.

```
<proxies>
    <proxy>
        <id>oracle-http-proxy</id>
        <host>xxxxx</host>
        <protocol>http</protocol>
        <nonProxyHosts>localhost|127.0.0.1|xxxxx</nonProxyHosts>
        <port>xxxxx</port>
        <active>true</active>
    </proxy>
</proxies>
```

8. Copy UI custom icons to directory older $WORKSPACEDIR/ata-builder/staging/downloads/ata-ui/images if you have any customizations for service topology icon. For making customizations, see "Customizing the Images".

Copyright © 2023, 2026, Oracle and/or its affiliates.

9. Update the image tag in $WORKSPACEDIR/ata-builder/bin/ata_manifest.yaml
10. Run `build-all-images` script to create ATA images:

```
$WORKSPACEDIR/ata-builder/bin/build-all-images.sh
```

> **Note**
>
> You can include the above procedure into your CI pipeline as long as the required components are already downloaded to the staging area.

Post-build Image Management

The ATA image builder creates images with names and tags based on the settings in the manifest file. By default, this results in the following images:

- uim-8.0.0.0.0-ata-base-2.0.0.0.0:latest
- uim-8.0.0.0.0-ata-api-2.0.0.0.0:latest
- uim-8.0.0.0.0-ata-pgx-2.0.0.0.0:latest
- uim-8.0.0.0.0-ata-ui-2.0.0.0.0:latest
- uim-8.0.0.0.0-ata-dbinstaller-2.0.0.0.0:latest
- uim-8.0.0.0.0-ata-consumer-2.0.0.0.0:latest
- uim-8.0.0.0.0-alarm-consumer-2.0.0.0.0:latest
- uim-8.0.0.0.0-smartsearch-consumer-2.0.0.0.0:latest
- uim-8.0.0.0.0-impact-analysis-api-2.0.0.0.0:latest

Customizing the Images

Service topology can be customized using a JSON configuration file. See "Customizing ATA Service Topology Configurations from UIM" for more information. As a part of customization, if custom icons are to be used to represent nodes in service topology, they must be placed in the **$WORKSPACEDIR/ata-builder/staging/downloads/ata-ui/images/** folder and **ata-ui image** must be rebuilt.

Localizing Specification Name in ATA

The specification names in the ATA application can be localized. To achieve this, export the specification bundle from UIM and build the image with customization.

The prerequisite for localizing specification name in ATA is that the specification displays names provided in Service Catalog and Design - Design Studio and deployed to UIM.

To localize the specification name:

1. Export UIM App Bundle:
   a. In UIM UI, navigate to **Execute Rule** under **Administration** section in the left pane.
   b. Select **EXPORT_SPECIFICATION_DISPLAY_NAMES_AS_JSON** from the dropdown, ignore the file upload option, and click **Process**.

Copyright © 2023, 2026, Oracle and/or its affiliates.

> **Note**
>
> The **ora_uim_baserulesets** must be deployed before performing this step.

c. Download **uimAppBundle.tar.gz**.

2. Place **uimAppBundle.tar.gz** in **$BUILDER_HOME/staging/downloads/ata-ui/uimAppBundle** builder for image building.
3. Customize and build the image.

Creating an ATA Instance

This section describes how to create an ATA instance in your cloud native environment using the operational scripts and the configuration provided in the common cloud native toolkit.

Before you can create an ATA instance, you must validate cloud native environment. See "Planning UIM Installation" for details on prerequisites.

In this section, while creating a basic instance, the project name is considered as **sr** and instance name is considered as **quick**.

> **Note**
>
> Project and Instance names cannot contain any special characters.

Installing ATA Cloud Native Artifacts and Toolkit

Build container images for the following using the ATA cloud native Image Builder:

- ATA Core application
- ATA PGX application
- ATA Consumer application
- Alarm Consumer application
- ATA User Interface application
- ATA database installer
- SmartSearch Consumer application

See "Deployment Toolkits" to download the Common cloud native toolkit archive file. Set the variable for the installation directory by running the following command,
where $WORKSPACEDIR is the installation directory of the COMMON cloud native toolkit:

```
export COMMON_CNTK=$WORKSPACEDIR/common-cntk
```

Setting up Environment Variables

ATA relies on access to certain environment variables to run seamlessly. Ensure the following variables are set in your environment:

- Path to your common cloud native toolkit

Copyright © 2023, 2026, Oracle and/or its affiliates.

- Path to your specification files

To set the environment variables:

1. Set the COMMON_CNTK variable to the path of directory where common cloud native toolkit is extracted as follows:

   ```
   $ export COMMON_CNTK=$WORKSPACEDIR/common-cntk
   ```

2. Set SPEC_PATH variable to the location where application and database yamls are copied. See "Assembling the Specifications" to copy specification files if not already copied.

   ```
   $ export SPEC_PATH=$WORKSPACEDIR/spec_dir
   ```

Creating Secrets

You must store sensitive data and credential information in the form of Kubernetes Secrets that the scripts and Helm charts in the toolkit consume. Managing secrets is out of the scope of the toolkit and must be implemented while adhering to your organization's corporate policies. Additionally, ATA service does not establish password policies.

> **Note**
>
> The passwords and other input data that you provide must adhere to the policies specified by the appropriate component.

As a prerequisite to use the toolkit for either installing the ATA database or creating a ATA instance, you must create secrets to access the following:

- ATA Database
- UIM Instance Credentials
- Common **oauthConfig** Secret for authentication
- **commonTrust** secret for egress SSL communication

The toolkit provides sample scripts to perform this. These scripts should be used for manual and faster creation of an instance. It does not support any automated process for creating instances. The scripts also illustrate both the naming of the secret and the layout of the data within the secret that ATA requires. You must create secrets before running the install-database.sh or create-applications.sh scripts.

Creating Secrets for ATA Database Credentials

The database secret specifies the connectivity details and the credentials for connecting to the ATA PDB (ATA schema). This is consumed by the ATA DB installer and ATA runtime.

> **Note**
>
> The ATA schema username and PGX client username should have uppercase letters. The lowercase letters are not allowed.

Copyright © 2023, 2026, Oracle and/or its affiliates.

1. Run the following script to create the required secrets:

```
$COMMON_CNTK/scripts/manage-app-credentials.sh -p sr -i quick -
s $SPEC_PATH -a ata create database
```

2. Enter the corresponding values as prompted:
   - TOPOLOGY DB Admin (sys) Username: Provide Topology Database admin username
   - TOPOLOGY DB Admin (sys) Password: Provide Topology Database admin password
   - TOPOLOGY Schema Username: Provide username for ATA schema to be created
   - TOPOLOGY Schema Password: Provide ATA schema password
   - TOPOLOGY DB Host: Provide ATA Database Hostname
   - TOPOLOGY DB Port: Provide ATA Database Port
   - TOPOLOGY DB Service Name: Provide ATA Service Name
   - PGX Client Username: Provide username for PGX Client User to be created
   - PGX Client Password: Provide PGX Client Password
3. Verify that the following secret is created:

```
sr-quick-ata-db-credentials
```

Creating Secrets for UIM Credentials

The UIM secret specifies the credentials for connecting to the UIM application. This is consumed by ATA runtime.

1. Run the following scripts to create the UIM secret:

```
$COMMON_CNTK/scripts/manage-app-credentials.sh -p sr -i quick -
s $SPEC_PATH -a ata create uim
```

2. Enter the credentials and the corresponding values as prompted. The credentials should be as shown in the following example:

```
Provide UIM credentials ...(Format should be http: //<host>:<port>)
UIM URL: Provide UIM Application URL, sample https://quick.sr.uim.org:30443
UIM Username: Provide UIM username
UIM Password: Provide UIM password
Is provided UIM a Cloud Native Environment ? (Select number from menu)
1) Yes
2) No
#? 1
Provide UIM Cluster Service name (Format <uim-project>-<uim-instance>-
cluster-uimcluster.<project>.svc.cluster.local)
UIM Cluster Service name: sr-quick-cluster-uimcluster.sr.svc.cluster.local
#Provide UIM Cluster Service name.
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

> **Note**
>
> Provide the default UIM URL, if the SAML protocol is configured for authentication with any IDP (such as IDCS, Keycloak, and so on.) For example: `https://<instance>.<project>.<hostSuffix>:<loadbalancerport>`.

3. Verify that the following secret is created:

   ```
   sr-quick-ata-uim-credentials
   ```

Creating Secrets for Authentication Server Details

The OAuth secret specifies details of the authentication server. It is used by ATA to connect to Message Bus. See "Adding Common OAuth Secret and ConfigMap" for more information.

If authentication is enabled on ATA, ensure that you create an **oauthConfig** secret with the appropriate OIDC details of your identity provider. To create an **oauthConfig** secret, see "Adding Common OAuth Secret and ConfigMap".

Secret to Pass Egress Certificates

Create **commonTrust** secret. This secret should contain truststore with all SSL certificates required for SSL egress communication. For example: Traditional UIM cert, Idp cert. and so on.

Configuring the Application Specification Files

The **applications-base.yaml**, **app-ata.yaml**, **<shape>/ata.yaml** files are Helm override values file to override default values of ATA chart. Update values at $SPEC_PATH/<PROJECT>/<INSTANCE> to override the default values.

The **applications-base.yaml** contains values that are common for all microservices. Provide Values under that common section and it is reflected for all services.

> **Note**
>
> There are common values specified in **applications-base.yaml** and **database.yaml** for the microservices. To override the common value, specify the value for the common value under the specific file (for example: app-ata.yaml) for service. If value under the chart is empty, then common value is considered.

To configure the application specification:

1. Edit the **applications-base.yaml** specification file as follows:
   a. Update `loadbalancerport`. If there is no external loadbalancer configured for the instance, change the value of `loadbalancerport` to the ingressController NodePort . If SSL is enabled on ATA, provide SSL NodePort and if SSL is disabled, provide non-SSL NodePort.

Copyright © 2023, 2026, Oracle and/or its affiliates.

If you use Oracle Cloud Infrastructure LBaaS or any other external load balancer and if TLS is enabled, set `loadbalancerport` to **443**. Else, set `loadbalancerport` to **80** and update the value for `loadbalancerhost` appropriately.

```
#provide loadbalancer port
loadbalancerport: 30505
```

b. Provide the `ingressController` details as below for HAproxy ingress controller. If you are using any other ingress controller, provide the corresponding details:

```
ingressController: "GENERIC"
ingress:
  #provide appropriate ingressClass for controller, “haproxy” is
default for haproxy ingressController.
  className: "haproxy"
  annotations:
    haproxy.org/cookie-persistence: "uimhaproxycookie"
```

c. To enable authentication, set the **authentication.enabled** flag to **true**. If you use any other IDP and it is not under the public DNS server, you can provide the **hostAliases**.

```
# The enabled flag is to enable or disable authentication
authentication:
enabled: true

hostAliases:
- ip: <ip-address> hostnames:
- <dns-name> ex. my.service.oracle.com
```

d. If your environment requires a password to download the container images from your repository, create a Kubernetes secret with the Docker pull credentials. See https://kubernetes.io/docs/tasks/configure-pod-container/pull-image-private-registry/#create-a-secret-by-providing-credentials-on-the-command-line for more information.

```
# The image pull access credentials for the "docker login" into Docker
repository, as a Kubernetes secret.
# uncomment and set if required.

#imagePullSecret:
#  imagePullSecrets:
#    - name: regcred
```

e. Select the shape to be used. The shape is the file that contains the resources for ATA service and is available at **$SPEC_PATH/$PROJECT/$INSTANCE/shape** directory.

```
shape: dev
```

2. Edit **app-ata.yaml** to provide ATA specific configurations as follows:

a. Provide the image in your repository (name and tag) by running the following command:

```
vi $SPEC_PATH/$PROJECT/$INSTANCE/app-ata.yaml

# edit the image names, to reflect the ATA image names and location in
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
your docker repository
# edit the image tags to reflect the ATA image names and location in
your docker repository

ata:
  name: "ata"
  image:
    topologyApiName: uim-<uim-version>-ata-api-<ata-version>
    pgxName: uim-<uim-version>-ata-pgx-<ata-version>
    uiName: uim-<uim-version>-ata-ui-<ata-version>
    topologyConsumerName: uim-<uim-version>-ata-consumer-<ata-version>
    smartsearchConsumerName: uim-<uim-version>-smartsearch-consumer-
<ata-version>
    alarmConsumerName: uim-<uim-version>-alarm-consumer-<ata-version>
    impactAnalysisApiName: uim-<uim-version>-impact-analysis-api-<ata-
version>
    topologyApiTag: latest
    pgxTag: latest
    uiTag: latest
    topologyConsumerTag: latest
    smartsearchConsumerTag: latest
    alarmConsumerTag: latest
    impactAnalysisApiTag: latest
    repository:
    repositoryPath:
```

b. Set Pull Policy for ATA images. Set `pullPolicy` to **Always** in case image is updated.

```
ata:
  image:
    pullPolicy: IfNotPresent
```

Installing ATA Service Schema

To install the ATA schema:

1. Update values under ata-dbinstaller in $SPEC_PATH/sr/quick/database.yaml file with values required for ATA schema creation.

> **Note**
>
> - The YAML formatting is case-sensitive. Use a YAML editor to ensure that you do not make any syntax errors while editing. Follow the indentation guidelines for YAML.
> - Before changing the default values provided in the specification file, verify that they align with the values used during PDB creation. For example, the default tablespace name should match the value used when PDB is created.

2. Edit the database.yaml file and update the DB installer image to point to the location of your image as follows:

```
ata-dbinstaller:
  dbinstaller:
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
    image:  DB_installer_image_in_your_repo
    tag: DB_installer image tag in your repo
```

3. If your environment requires a password to download the container images from your repository, create a Kubernetes secret with the Docker pull credentials. See "Kubernetes documentation" for details. Refer the secret name in the **database.yaml**. Provide image pull secret and image pull policy details.

   ```
   ata-dbinstaller:
     imagePullPolicy: Never
   # The image pull access credentials for the "docker login" into Docker
   repository, as a Kubernetes secret.
   # Uncomment and set if required.
   #  imagePullSecret: ""
   ```

4. Run the following script to start the ATA DB installer, which instantiates a Kubernetes pod resource. The pod resource lives until the DB installation operation completes.

   ```
   $COMMON_CNTK/scripts/install-database.sh -p sr -i quick -s $SPEC_PATH -a
   ata -c 1
   ```

5. You can run the script with -h to see the available options.
6. Check the console to see if the DB installer is installed successfully.
7. If the installation has failed, run the following command to review the error message in the log:

   ```
   kubectl logs -n sr sr-quick-ata-dbinstaller
   ```

8. Clear the failed pod by running the following command:

   ```
   helm uninstall sr-quick-ata-dbinstaller -n sr
   ```

9. Run the install-database script again to install the ATA DB installer.

Configuring ATA

Sample configuration files topology-static-config.yaml.sample, topology-dynamic-config.yaml.sample are provided as the sample files for ATA API service that are under **$SPEC_PATH/$PROJECT/$INSTANCE/config/ata/**.

> **Note**
>
> If you do not see these files at the above location, make sure that you run **assemble-specification** script.

To override configuration properties, copy the sample static property file to **topology-static-config.yaml** and sample dynamic property file to **topology-dynamic-config.yaml**. Provide key value to override the default value provided out-of-the-box for any specific system configuration property. The properties defined in property files are fed into the container using Kubernetes configuration maps. Any changes to these properties require the instance to be upgraded. Pods are restarted after configuration changes to **topology-static-config.yaml**.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Max Rows

Modify the following setting to limit the number of records returned in LIMIT queries:

```
topology:
    query:
      maxrows: 5000
```

Date Format

Any modifications to the date format used by all dates must be consistently applied to all consumers of the APIs.

```
topology:
    api:
      dateformat: yyyy-MM-dd'T'HH:mm:ss.SSS'Z'
```

Alarm Types

The out of the box alarm types utilize industry standard values. If you want to display a different value, modify the value accordingly:

For example: To modify the COMMUNICATIONS_ALARM change the value to COMMUNICATIONS_ALARM: Communications

```
alarm-types:
      COMMUNICATIONS_ALARM: COMMUNICATIONS_ALARM
      PROCESSING_ERROR_ALARM: PROCESSING_ERROR_ALARM
      ENVIRONMENTAL_ALARM: ENVIRONMENTAL_ALARM
      QUALITY_OF_SERVICE_ALARM: QUALITY_OF_SERVICE_ALARM
      EQUIPMENT_ALARM: EQUIPMENT_ALARM
      INTEGRITY_VIOLATION: INTEGRITY_VIOLATION
      OPERATIONAL_VIOLATION: OPERATIONAL_VIOLATION
      PHYSICAL_VIOLATION: PHYSICAL_VIOLATION
      SECURITY_SERVICE: SECURITY_SERVICE
      MECHANISM_VIOLATION: MECHANISM_VIOLATION
      TIME_DOMAIN_VIOLATION: TIME_DOMAIN_VIOLATION
```

Event Status

ATA supports 3 types of events: 'Raised' for new events, 'Updated' for existing events with updated information and 'Cleared' for events that have been Closed.

To modify the 'CLEARED' event change the value to CLEARED: closed

```
    event-status:
      CLEARED: CLEARED
      RAISED: RAISED
      UPDATED: UPDATED
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

Event Severity

ATA supports various types of event severity on a Device. The severity from most severe to least severe is CRITICAL(1), MAJOR(5), WARNING(10), INTERMEDIATE(15), MINOR(20), CLEARED(25) and None(999).

Internally, a numeric value is used to identify the severity hierarchy. The top three most severe events are tracked in ATA.

To modify the 'INTERMEDIATE' severity change the value to INTERMEDIATE: moderate

```
severity:
      CLEARED: CLEARED
      INDETERMINATE: INDETERMINATE
      CRITICAL: CRITICAL
      MAJOR: MAJOR
      MINOR: MINOR
      WARNING: WARNING
```

Path Analysis Cost Values

ATA supports 3 different types of numeric cost values for each edge/connectivity maintained in topology. The cost type label is configured based on your business requirements and data available.

You select the cost parameter to evaluate while using path analysis. The cost values are maintained externally using the REST APIs.

To modify 'costValue3' from Distance to Packet Loss change the value to costValue3: PacketLoss after updating the data values.

```
pathAnalysis:
  costType:
    costValue1: Jitter
    costValue2: Latency
    costValue3: Distance
```

Path Analysis Alarms

Alarms can be used by path analysis to exclude devices in the returned paths. The default setting is to exclude devices with any alarm.

To allow Minor and Greater alarms modify the setting to:

excludeAlarmTypes: Critical and Greater, Major and Greater

All Paths Limit

To improve the response time, modify the max number of paths returned when using 'All' Paths.

Configuring Topology Consumer

The sample configuration files **topology-static-config.yaml.sample** and **topology-dynamic-config.yaml.sample** are provided under **$SPEC_PATH/$PROJECT/$INSTANCE/config/ata** .

Copyright © 2023, 2026, Oracle and/or its affiliates.

> **Note**
>
> If you do not see these files, make sure that you run the **assemble-specifications** script.

To override configuration properties, copy the sample static property file to **topology-static-config.yaml** and sample dynamic property file to **topology-dynamic-config.yaml**. Provide key value to override the default value provided out-of-the-box for any specific system configuration property. The properties defined in property files are provided to the container using Kubernetes configuration maps. Any changes to these properties require the instance to be upgraded. Pods are restarted after configuration changes to **topology-static-eexconfig.yaml**.

Reduce the Poll size for Retry and dlt Topic

Uncomment or add the configuration values in **topology-config.yaml** and upgrade the Topology Consumer service.

Maximum Poll Interval and Records

Edit **max.poll.interval.ms** to increase or decrease the delay between invocations of **poll()** when using consumer group management and **max.poll.records** to increase or decrease the maximum number of records returned in a single call to **poll()**.

```
mp.messaging:
      incoming:
        toInventoryChannel:
    #      max.poll.interval.ms: 300000
    #      max.poll.records: 500
        toFaultChannel:
    #      max.poll.interval.ms: 300000
    #      max.poll.records: 500
        toRetryChannel:
    #      max.poll.interval.ms: 300000
    #      max.poll.records: 200
        toDltChannel:
    #      max.poll.interval.ms: 300000
    #      max.poll.records: 100
```

Partition assignment strategy

The **PartitionAssignor** is the class that decides which partitions are assigned to which consumer. While creating a new Kafka consumer, you can configure the strategy that can be used to assign the partitions amongst the consumers. You can set it using the configuration **partition.assignment.strategy**. The partition re-balance (moving partition ownership from one consumer to another) happens, in case of:

- Addition of new Consumer to the Consumer group.
- Removal of Consumer from the Consumer group.
- Addition of New partition to the existing topic.

To change the partition assignment strategy, update the *topology-config.yaml* for topology consumer and redeploy the POD. The below example configuration shows the

Copyright © 2023, 2026, Oracle and/or its affiliates.

CooperativeStickyAssignor strategy. For list of supported partition assignment strategies, see **partition.assignment.strategy** in Apache Kafka documentation.

```
mp.messaging
  connector:
    helidon-kafka:
      partition.assignment.strategy:
org.apache.kafka.clients.consumer.CooperativeStickyAssignor
```

Configuring SmartSearch Consumer

The sample configuration files **smartsearch-consumer-static-config.yaml.sample** and **smartsearch-consumer-dynamic-config.yaml.sample** are provided under **$SPEC_PATH/$PROJECT/$INSTANCE/config/smartsearch-consumer**.

> **Note**
>
> If you do not see these files, make sure that you run the **assemble-specifications** script.

To override configuration properties, copy the sample static property file to **smartsearch-consumer-static-config.yaml** and sample dynamic property file to **smartsearch-consumer-dynamic-config.yaml**. Provide key value to override the default value provided out-of-the-box for any specific system configuration property. The properties defined in property files are provided to the container using Kubernetes configuration maps. Any changes to these properties require the instance to be upgraded. Pods are restarted after configuration changes to **smartsearch-consumer-static-config.yaml**.

Reduce the Poll size for Retry and dlt Topic

Uncomment or add the configuration values in **smartsearch-consumer-config.yaml** and upgrade the smartsearch consumer service.

Maximum Poll Interval and Records

Edit **max.poll.interval.ms** to increase or decrease the delay between invocations of **poll()** when using consumer group management and **max.poll.records** to increase or decrease the maximum number of records returned in a single call to **poll()**.

```
mp.messaging:
      incoming:
        toInventoryChannel:
    #      max.poll.interval.ms: 300000
    #      max.poll.records: 500
        toFaultChannel:
    #      max.poll.interval.ms: 300000
    #      max.poll.records: 500
        toRetryChannel:
    #      max.poll.interval.ms: 300000
    #      max.poll.records: 200
        toDltChannel:
    #      max.poll.interval.ms: 300000
    #      max.poll.records: 100
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

Partition assignment strategy

The **PartitionAssignor** is the class that decides which partitions are assigned to which consumer. While creating a new Kafka consumer, you can configure the strategy that can be used to assign the partitions amongst the consumers. You can set it using the configuration **partition.assignment.strategy**. The partition re-balance (moving partition ownership from one consumer to another) happens, in case of:

- Addition of new Consumer to the Consumer group.
- Removal of Consumer from the Consumer group.
- Addition of New partition to the existing topic.

To change the partition assignment strategy, update the **smartsearch-consumer-static-config.yaml** for SmartSearch consumer and redeploy the POD. The below example configuration shows the **CooperativeStickyAssignor** strategy. For list of supported partition assignment strategies, see **partition.assignment.strategy** in Apache Kafka documentation.

```
mp.messaging
  connector:
    helidon-kafka:
      partition.assignment.strategy:
org.apache.kafka.clients.consumer.CooperativeStickyAssignor
```

Integrate ATA Service with Message Bus Service

To integrate ATA API service with Message Bus service:

1. In the file **$SPEC_PATH/sr/quick/applications-base.yaml**, uncomment the section **messagingBusConfig**.
2. Provide namespace and instance name on which the Messaging Bus service is deployed.
3. Security protocol is SASL_PLAINTEXT if authentication is enabled on Message bus service. If authentication is not enabled on the Message Bus service, the security protocol is PLAINTEXT.

A sample configuration when authentication is enabled and Messaging Bus is deployed on instance 'quick' and namespace 'sr' is as follows:

applications-base.yaml

```
authentication:
  enabled: true

messagingBusConfig:
  namespace: sr
  instance: quick
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

Integrating ATA with Authorization Service

If Authorization Service is installed, you can integrate ATA with Authorization service by updating **$SPEC_PATH/$PROJECT/$INSTANCE/applications-base.yaml** and uncommenting the following properties and provide appropriate values:

```
authorizationServiceConfig:
  namespace: <project> #namespace on which authorization service is deployed
  instance: <instance> #instance name on which authorization service is
deployed
```

Creating an ATA Instance

To create an ATA instance in your environment using the scripts that are provided with the toolkit:

1. Run the following command to create an ATA instance:

   ```
   $COMMON_CNTK/scripts/create-applications.sh -p sr -i quick -s $SPEC_PATH -
   a ata
   ```

   The create-applications script uses the helm chart located in $COMMON_CNTK/charts/ata-app to create and deploy a ata service.
2. Run the following command to create an ATA Ingress:

   ```
   $COMMON_CNTK/scripts/create-ingress.sh -p sr -i quick -s $SPEC_PATH -a ata
   ```

   The **create-ingress** script creates an Ingress object that defines hostname and path-based routing rules to direct incoming requests to ATA services through the Ingress Controller. It is registered with the Ingress Controller based on the **className** property specified in the **applications-base.yaml** file.
3. If the scripts fail, see the Troubleshooting Issues section at the end of this topic, before you make additional attempts.

For more information on creating an ATA instance, see "Creating an ATA Instance"

Accessing ATA Instance

Proxy Settings

To set the proxy settings:

1. In the browser's network no-proxy settings include ***<hostSuffix>**. For example, `*uim.org`.
2. In `/etc/hosts` include **etc/hosts**

   ```
   <k8s cluster ip or loadbalancerIP>
   <instance>.<project>.topology.<hostSuffix>

   for example: <k8s cluster ip or external loadbalancer ip>
   quick.sr.topology.uim.org
   ```


Copyright © 2023, 2026, Oracle and/or its affiliates.

Exercise ATA service endpoints

If TLS is enabled on ATA, exercise endpoints using Hostname <topology-instance>.<topology-project>.topology.uim.org.

ATA UI endpoint format: https://<topology-instance>.<topology-project>.topology.<hostSuffix>:<port>/apps/ata-ui

ATA API endpoint format: https://<topology-instance>.<topology-project>.topology.<hostSuffix>:<port>/topology/v2/vertex

- ATA UI endpoint: https://quick.sr.topology.uim.org:30443/apps/ata-ui
- ATA API endpoint: https://quick.sr.topology.uim.org:30443/topology/v2/vertex

If TLS is not enabled on ATA, exercise endpoints:

ATA UI endpoint format: http://<topology-instance>.<topology-project>.topology.<hostSuffix>:<port>/apps/ata-ui

ATA API endpoint format: http://<topology-instance>.<topology-project>.topology.<hostSuffix>:<port>/topology/v2/vertex

Validating the ATA Instance

To validate the ATA instance:

1. Run the following to check the status of ata instance deployed.

   ```
   $COMMON_CNTK/scripts/application-status.sh -p sr -i quick -s $SPEC_PATH -a ata
   ```

   The application-status script returns the status of ATA service deployments and pods status.
2. Run the following endpoint to monitor health of ata:

   ```
   https://quick.sr.topology.uim.org:<loadbalancerport>/health
   ```

3. Run the following ATA service endpoints to add entry in /etc/hosts <k8s cluster ip or external loadbalancer ip> quick.sr.topology.uim.org:
   - ATA UI endpoint: `https://quick.sr.topology.uim.org:30443/apps/ata-ui`
   - ATA API endpoint: `https://quick.sr.topology.uim.org:30443/topology/v2/vertex`

Deploying the Graph Server Instance

Graph Server or Pgx Server instance is needed for Path Analysis. By default, replicaCount of pgx(graph) server pods is set to '0'. For path analysis to function , set the replicaCount of pgx pods to '2' and upgrade instance. See "Upgrading the ATA Instance" for more information.

A **cron** job must be scheduled to periodically reload the active ata-pgx pod.

```
pgx:
  pgxName: "ata-pgx"
  replicaCount: 2
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
  java:
    user_mem_args: "-Xms8000m -Xmx8000m -XX:+HeapDumpOnOutOfMemoryError -
XX:HeapDumpPath=/logMount/$(APP_PREFIX)/ata/ata-pgx/"
    gc_mem_args: "-XX:+UseG1GC"
    options:
  resources:
    limits:
      cpu: "4"
      memory: 16Gi
    requests:
      cpu: 3500m
      memory: 16Gi
```

> **Note**
>
> If the PGX pod is required, verify the following database parameters before starting the pod. Any changes to these parameters require a database restart:
>
> - **undo_retention** (recommended value: 1800)
> - **db_flashback_retention_target** (default value: 1440 minutes)

Scheduling the Graph Server Restart CronJob

Once the instance is created succesfully, cronjob needs to schedule for ata-pgx pod restarts. For a scheduled period of time, one of the ata-pgx pod is restarted and all incoming requests are routed to other unfified-topology-pgx pod seamlessly.

Update the script $COMMON_CNTK/samples/cronjob-scripts/pgx-restart.sh to include required environment variables - KUBECONFIG, pgx_ns, pgx_instance. For a basic instance, pgx_ns is sr and pgx_instance is quick.

```
export KUBECONFIG=<kube config path>
export pgx_ns=<ata project name>
export pgx_instance=<ata instance name>
pgx_pods=`kubectl get pods -n $pgx_ns --sort-by=.status.startTime -o name |
awk -F "/" '{print $2}' | grep $pgx_instance-ata-pgx`
pgx_pod_arr=( $pgx_pods )
echo "Deleting pod - ${pgx_pod_arr[0]}"
kubectl delete pod ${pgx_pod_arr[0]} -n $pgx_ns --grace-period=0
```

The following crontab is scheduled for every day midnight. Scheduled time may vary depending on the volume of data.

Variable $COMMON_CNTK should be set in environment where cronjob runs or replace $COMMON_CNTK with complete path.

```
crontab –e 0 0 * * * $COMMON_CNTK/samples/cronjob-scripts/pgx-restart.sh
> $COMMON_CNTK/samples/cronjob-scripts/pgx-restart.log
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

Affinity on Graph Server

If multiple PGX pods are scheduled on the same worker node, the memory consumption by these PGX pods becomes very high. To address this, include the following affinity rule in **app-ata.yaml**, under the ata chart to avoid scheduling of multiple PGX pods on the same worker node.

The following **podantiaffinity** rule uses the **app= <topology-project>-<topology-instance>-ata-pgx** label. Update the label with the corresponding project and instance names for ATA service. For example: **sr-quick-ata-pgx**.

```
ata:
  affinity:
    podAntiAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        - labelSelector:
            matchExpressions:
              - key: app
                operator: In
                values:
                - <topology-project>-<topology-instance>-ata-pgx
          topologyKey: "kubernetes.io/hostname"
```

Upgrading the ATA Instance

Upgrading ATA is required when there are updates made to **applications-base.yaml**, **app-ata.yaml**, **<shape>/ata.yaml**, and **topology-static-config.yaml** and **topology-dynamic-config.yaml** configuration files.

Run the following command to upgrade ATA service.

```
$COMMON_CNTK/scripts/upgrade-applications.sh -p sr -i quick -s $SPEC_PATH -a
ata
```

After script execution is done, validate the ATA service by running application-status script.

Restarting the ATA Instance

To restart the ATA instance:

1. Run the following command to restart ATA service

   ```
   $COMMON_CNTK/scripts/restart-applications.sh -p sr -i quick -s $SPEC_PATH -
   a ata -r all
   ```

2. After running the script, validate the ATA service by running application-status script.
3. To restart **ata-api/ata-ui/ata-pgx/ata-consumer/alarm-consumer/smartsearch-consumer**, run the above command by passing **-r** with service name as follows:

Copyright © 2023, 2026, Oracle and/or its affiliates.

4. To restart ATA API

```
$COMMON_CNTK/scripts/restart-applications.sh -p sr -i quick -s $SPEC_PATH -
a ata -r ata-api
```

5. To restart ATA PGX

```
$COMMON_CNTK/scripts/restart-applications.sh -p sr -i quick -s $SPEC_PATH -
a ata -r ata-pgx
```

6. To restart ATA UI:

```
$COMMON_CNTK/scripts/restart-applications.sh -p sr -i quick -s $SPEC_PATH -
a ata -r ata-ui
```

7. To restart ATA Consumer:

```
$COMMON_CNTK/scripts/restart-applications.sh -p sr -i quick -s $SPEC_PATH -
a ata -r ata-consumer
```

8. To restart Alarm Consumer:

```
$COMMON_CNTK/scripts/restart-applications.sh -p sr -i quick -s $SPEC_PATH -
a ata -r alarm-consumer
```

9. To restart SmartSearch Consumer:

```
$COMMON_CNTK/scripts/restart-applications.sh -p sr -i quick -s $SPEC_PATH -
a ata -r smartsearch-consumer
```

10. To restart Service Impact Analysis:

```
$COMMON_CNTK/scripts/restart-applications.sh -p sr -i quick -s $SPEC_PATH -
a ata -r impact-analysis
```

Deleting and Recreating a ATA Instance

- Run the following command to delete the ATA service:

```
$COMMON_CNTK/scripts/delete-applications.sh -p sr -i quick -s $SPEC_PATH -
a ata
```

- Run the following command to delete the ATA schema:

```
$COMMON_CNTK/scripts/install-database.sh -p sr -i quick -s $SPEC_PATH -a
ata -c 2
```

- Run the following command to create the ATA schema:

```
$COMMON_CNTK/scripts/install-database.sh -p sr -i quick -s $SPEC_PATH -a
ata -c 1
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

- Run the following command to create the ATA service:

```
$COMMON_CNTK/scripts/create-applications.sh -p sr -i quick -s $SPEC_PATH -
a ata
```

Alternate Configuration Options for ATA

You can configure ATA using the following alternate options.

Setting up Secure Communication using TLS

When ATA service is involved in secure communication with other systems, either as the server or as the client, you should additionally configure SSL/TLS.The procedures for setting up TLS use self-signed certificates for demonstration purposes. However, replace the steps as necessary to use signed certificates. To generate common self-signed certificates, see "SSL Certificates".

To setup secure communication using TLS:

1. Edit the **$SPEC_PATH/sr/quick/applications-base.yaml** and set **tls enabled** to **true**.

```
tls:
    enabled: true
```

2. Specify the SSL port of loadbalancer or IngressController in **applications-base.yaml**. You use this port to access the application from outside:

```
loadbalancerport: 30543
```

3. Create the **ingressTLS** secret to pass the generated certificate and key pem files.

```
$COMMON_CNTK/scripts/manage-app-credentials.sh -p sr -i quick -
s $SPEC_PATH -a ata create ingressTLS
```

4. The script prompts for the following detail:
   a. Ingress TLS Certificate Path (PEM file): <path_to_commoncert.pem>
   b. Ingress TLS Key file Path (PEM file): <path_to_commonkey.pem>
5. Verify that the following secrets are created successfully.

```
sr-quick-ata-ingress-tls-cert-secret
```

6. Create ATA instance and Ingress as usual. Access ATA endpoints using hostname <topology-instance>.<topology-instance>.topology.uim.org:

```
#If instance already running upgrade instance, and delete and create
Ingress.

#instance
$COMMON_CNTK/scripts/upgrade-applications.sh -p sr -i quick -s $SPEC_PATH -
a ata

#ingress
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
$COMMON_CNTK/scripts/delete-ingress.sh -p sr -i quick -s $SPEC_PATH -a ata
$COMMON_CNTK/scripts/create-ingress.sh -p sr -i quick -s $SPEC_PATH -a ata
```

7. Add entry in /etc/hosts <k8s cluster ip or external loadbalancer ip> quick.sr.topology.uim.org
8. ATA UI endpoint: `https://quick.sr.topology.uim.org:30443/apps/ata-ui`
9. ATA API endpoint: `https://quick.sr.topology.uim.org:30443/topology/v2/vertex`

Supporting the Wildcard Certificate

Smartsearch supports wildcard certificates. You can generate the **wildCard** certificates with the **hostSuffix** value provided in **applications-base.yaml**. The default is **uim.org**.

You must change the **subDomainNameSeperator** value from period(**.**) to hyphen(**-**) so that the incoming hostnames match the wild card DNS pattern.

Make the following updates to the **$SPEC_PATH/$PROJECT/$INSTANCE/applications-base.yaml file**.

```
#Uncooment and provide the value of subDomainNameSeparator, default is "."
#Value can be changed as "-" to match wild-card pattern of ssl certificates.
#Example hostnames for "-" quick-sr-topology.uim.org
subDomainNameSeparator: "-"
```

Using Annoation-Based Generic Ingress Controller

ATA supports standard Kubernetes ingress API and has samples for integration. In the following configuration, the required annotations for ATA for HAProxy, are provided.

Any Ingress Controller, which conforms to the standard Kubernetes ingress API and supports annotations required by ATA should work, although Oracle does not certify individual Ingress controllers to confirm this **generic** compatibility.

To use annotation-based generic ingress controller:

1. Update applications-base.yaml to provide the following annotations that enable stickiness through cookies:

```
# Valid values is GENERIC
ingressController: "GENERIC"

ingress:
  #provide appropriate ingressClass for controller, “haproxy” is default
for haproxy ingressController.
  className: "haproxy"
  annotations:
    haproxy.org/cookie-persistence: "uimhaproxycookie"
```

Enabling Authentication for ATA

This section provides you with information on enabling authentication for ATA.

The samples, for using IDCS as Identity Provider, are packaged with ATA. To use any Identity Provider of your choice, you must follow the corresponding configuration instructions.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Registering ATA in Identity Provider

You can register ATA as a Confidential application in Identity Provider. To do so:

1. Access the IDCS console and log in as administrator.
2. Navigate to the **Domains** and select the domain (*Default domain*) to add Helidon application as Confidential application.
3. Click **Add application** to register Helidon application as Confidential application.
   a. Choose **Confidential Application** and click **Launch workflow**.
   b. Enter the name as **ATA Application** and description as **ATA Application**.
   c. Select **Enforce grants as authorization** checkbox under **Authentication and authorization** section.
   d. Click **Next** at the bottom of the page.
   e. Choose **Configure this application as a resource server now** radio button under **Resource server configuration**.
   f. Enter **Primary Audience** as **https://<topology-hostname>:<loadbalancer-port>/**.
   g. Select **Add secondary audience** and enter IDCS URL as **Secondary audience**.
   h. Select **Add scopes** and add **ataScope** as **allowed scope**.
   i. Select **Configure this application as a client now** radio button under the **Client configuration** section.
   j. Select **Resource owner**, **Client credentials**, and **Authorization code** check boxes.
   k. Select **Allow HTTP URLs** check box only if your ATA application is not SSL enabled.
   l. Enter the following **Redirect URLs**:
      - https://<ata-hostname>:<loadbalancer-port>/topology
      - https://<ata-hostname>:<loadbalancer-port>/redirect/ata-ui/
      - https://<ata-hostname>:<loadbalancer-port>/sia
   m. Enter **Post-logout redirect URL** as **https://<ata-hostname>:<loadbalancer-port>/apps/ata-ui** (provide your Helidon application's home page URL).
   n. Enter **Logout URL** as **https://<ata-hostname>:<loadbalancer-port>/oidc/logout** (provide your Helidon application's logout URL).
   o. (Optional) Select **Bypass consent** button for skipping the consent page after IDCS login.
   p. Select **Anywhere** radio button for **Client IP address**.
   q. Click **Next** and click **Finish**.
4. Click **Activate** to create application (ATA Application).
5. Click **Activate application** from the pop-up window.
6. Click **Users** on the left side pane to assign users.
   a. Click **Assign users** to add domain users to the registered application.
   b. Choose the desired users from the pop-up window and click **Assign**.
7. (Optional) Click **Groups** on the left-side pane to assign groups.
   a. Click **Assign groups** to add domain groups to the registered application.

Copyright © 2023, 2026, Oracle and/or its affiliates.

b. Choose the desired groups from the pop-up window and click **Assign**.

> **Note**
>
> Make sure that the access token timeout (or IDP session timeout) is configured as per the requirement. If the access token gets timed out, the application gets logged out and user needs to login again.

Common Secret and Properties

You create a secret and config map with OAuth client details, which will be required for Message Bus and ATA.

Getting Client Credentials

You can get client credential details by navigating to your Oauth client on IDP. In case of IDCS, you can follow these steps to get the details.

Access the IDCS console and log in as Administrator. To get client credentials:

1. Navigate to **Domains** and select the domain (*Default domain*) to add Helidon application as Confidential application.
2. Click on the **ATA Application** name from the table.
3. Scroll to view the **Client secret** under the **General Information** section.
4. Click **Show secret** link to open a pop-up window showing the client secret.
5. Copy the link and store it to use in the Helidon application configuration.

Creating the OAuth Secrets and ConfigMap

To create **OauthConfig** secret with OIDC, see "Adding Common OAuth Secret and ConfigMap".

The sample for IDCS is as follows:

```
Identity Provider Uri: https://idcs-
df3063xxxxxxxxxx.identity.pint.oc9qadev.com:443
Client Scope: https://quick.sr.topology.uim.org:30443/first_scope
Client Audience: https://quick.sr.topology.uim.org:30443/
Token Endpoint Uri: https://idcs-
df3063xxxxxxxx.identity.pint.oc9qadev.com:443/oauth2/v1/token
Valid Issue Uri: https://identity.oraclecloud.com/
Introspection Endpoint Uri: https://idcs-
df3063xxxxxxx.identity.pint.oc9qadev.com:443/oauth2/v1/introspect
JWKS Endpoint Uri: https://idcs-
df3063xxxxxxxxxx.identity.pint.oc9qadev.com:443/admin/v1/SigningCert/jwk
Cookie Name: OIDCS_SESSION
Cookie Encryption Password: lpmaster
Provide Truststore details ...
Certificate File Path (ex. idpcert.pem): ./identity-pint-oc9qadev-com.pem
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

> **Note**
>
> For more details on IDCS, see "Common Configuration Options For all Services".

Choosing Worker Nodes for ATA Service

By default, ATA has its pods scheduled on all worker nodes in the Kubernetes cluster in which it is installed. However, in some situations, you may want to choose a subset of nodes where pods are scheduled.

For example:

Limitation on the deployment of ATA on specific worker nodes per each team for reasons such as capacity management, chargeback, budgetary reasons, and so on.

To choose a subset of nodes where pods are scheduled, you can use the configuration in the **app-ata.yaml** file.

Sample node affinity configuration(requiredDuringSchedulingIgnoredDuringExecution) for ATA service:

app-ata.yaml

```
ata:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: name
            operator: In
            values:
            - south_zone
```

Kubernetes pod is scheduled on the node with label name as *south_zone*. If node with label name: *south_zone* is not available, pod will not be scheduled.

Sample node affinity configuration (preferredDuringSchedulingIgnoredDuringExecution:) for ATA service:

app-ata.yaml

```
ata:
  affinity:
    nodeAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
          matchExpressions:
          - key: name
            operator: In
            values:
            - south_zone
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

Kubernetes pod is scheduled on the node with label name as *south_zone*. If node with label name: *south_zone* is not available, pod will still be scheduled on another node.

Setting up Persistent Storage

Follow the instructions mentioned in *UIM Cloud Native Deployment guide* for configuring Kubernetes persistent volumes.

To create persistent storage:

1. Update **applications-base.yaml** to enable storage volume for ATA service and provide the persistent volume name.

   ```
   storageVolume:
     enabled: true
     pvc: sr-nfs-pvc #Specify the storage-volume name
   ```

2. Update database.yaml to enable storage volume for ATA dbinstaller and provide the persistent volume name.

   ```
   storageVolume:
     enabled: true
     type: pvc
     pvc: sr-nfs-pvc #Specify the storage-volume name
   ```

After the instance is created, you must see the directories **ata** and **ata-dbinstaller** in your PV mount point, if you have enabled logs.

Managing ATA Logs

To customize and enable logging, update the logging configuration files for the application.

1. Customize ata-api service logs:
   - For service level logs update file $SPEC_PATH/$PROJECT/$INSTANCE/config/ata/ata-api/logging-config.xml
   - For Helidon-specific logs update file $SPEC_PATH/$PROJECT/$INSTANCE/config/ata/ata-api/logging.properties. By default console handler is used, you can provide filehandler as well uncomment below lines and provide <project> and <instance> names for location to save logs

     ```
     handlers=io.helidon.common.HelidonConsoleHandler,java.util.logging.FileH
     andler
     java.util.logging.FileHandler.formatter=java.util.logging.SimpleFormatte
     r
     java.util.logging.FileHandler.pattern=/logMount/sr-quick/ata/ata-api/
     logs/TopologyJULMS-%g-%u.log
     ```

2. Customize ata-pgx service logs:
   Update file $SPEC_PATH/$PROJECT/$INSTANCE/config/ata/pgx/logging-config.xml
3. Customize ata-ui service logs:
   Update file $SPEC_PATH/$PROJECT/$INSTANCE/config/ata/ata-ui/logging.properties

Copyright © 2023, 2026, Oracle and/or its affiliates.

4. Update the logging configuration files and upgrade the ata m-s application:

   ```
   $COMMON_CNTK/scripts/upgrade-applications.sh -p sr -i quick -s $SPEC_PATH -
   a ata
   ```

5. Customize the **ata-alarm-consumer** service log as follows:
   - For service-level logs update file:

     ```
     $SPEC_PATH/$PROJECT/$INSTANCE/config/ata/alarm-consumer/logging-
     config.xml
     ```

   - For Helidon-specific logs update file:

     ```
     $SPEC_PATH/$PROJECT/$INSTANCE/config/ata/alarm-consumer/
     logging.properties.
     ```

6. Customize SmartSearch Consumer service log:
   - For service level logs update the **$SPEC_PATH/$PROJECT/$INSTANCE/config/ata/smartsearch-consumer/logging-config.xml** file.
   - For Helidon-specific logs update the **$SPEC_PATH/$PROJECT/$INSTANCE/config/ata/smartsearch-consumer/logging.properties** file.

Viewing Logs using OpenSearch

You can view and analyze the Application logs using OpenSearch.

The logs are generated as follows:

1. Fluentd collects the application logs that are generated during cloud native deployments and sends them to OpenSearch.
2. OpenSearch collects all types of logs and converts them into a common format so that OpenSearch Dashboard can read and display the data.
3. OpenSearch Dashboard reads the data and presents it in a simplified view.

See "Deleting the OpenSearch and OpenSearch Dashboard Service" for more information.

Setting up FluentD

To enable centralized logging in Kubernetes using FluentD daemonset:

1. Run the following command to create a namespace and ensure that it does not exist already:

   ```
   kubectl get namespaces
   export FLUENTD_NS=fluentd
   kubectl create namespace $FLUENTD_NS
   ```

2. Update **$COMMON_CNTK/samples/charts/fluentd/charts/values.yaml** with OpenSearch **Host** and **Port**:

   ```
    opensearch:

        host: "opensearchHost"
   ```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
    port: "opensearchPort"

modify fluentd image, resources if required.

  image: fluent/fluentd-kubernetes-daemonset:v1.17-debian-opensearch-1

  resources:
    limits:
    memory: 200Mi
  requests:
    cpu: 100m
    memory: 200Mi
```

3. Update **$COMMON_CNTK/samples/charts/fluentd/charts/fluentd/templates/fluentd-config-map.yaml** by uncommenting and updating the username, password, and schema for OpenSearch connectivity.
4. Run the following commands to install **fluentd-logging** using the **$COMMON_CNTK/samples/charts/fluentd/values.yaml** file in the samples:

```
helm install fluentd-logging $COMMON_CNTK/samples/charts/fluentd -
n $FLUENTD_NS --values $COMMON_CNTK/samples/charts/fluentd/values.yaml \
--set namespace=$FLUENTD_NS \
--atomic --timeout 800s
```

5. Run the following command to upgrade fluentd-logging:

```
helm upgrade fluentd-logging $COMMON_CNTK/samples/charts/fluentd -
n $FLUENTD_NS --values $COMMON_CNTK/samples/charts/fluentd/values.yaml \
    --set namespace=$FLUENTD_NS \
    --atomic --timeout 800s
```

6. Run the following command to uninstall fluentd-looging:

```
helm delete fluentd-logging -n $FLUENTD_NS
```

7. Use `fluentd_logging-*` (default index config) index pattern in OpenSearch Dashboard to check the logs.

Configuring Shape for ATA

The predefined shapes: **devsmall**, **dev**, **prodsmall**, **prod**, and **prodlarge** are provided and the corresponding configurations are available at location **$SPEC_PATH/$PROJECT/$INSTANCE/shapes/**. To use a specific shape, specify the name in the **applications-base.yaml** file. You can also create and customize your own shapes. See "Customizing the Shapes" for detailed instructions. If you want to use a different shape for Messaging Bus, specify the **shape** value in the **app-ata.yaml** file.

Managing ATA Metrics

Run the following endpoint to monitor metrics of ATA:

```
https://instance.project.topology.<hostSuffix>:<loadbalancerport>/metrics
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

Prometheus and Grafana setup

See "Setting Up Prometheus and Grafana" for more information.

Configuring Metrics for Prometheus

See "Configuring Metrics for Services" for more information.

Allocating Resources for ATA Service Pods

To increase performance of the service, **<shape>/ata.yaml** has configuration to provide JVM memory settings and pod resources for ATA Service.

There are separate configurations provided for ata-api, topology-consumer, alarm-consumer, smartsearch-consumer, pgx, and ata-ui services. Provide required values under the service name under ata application.

```
ata:
  topologyApi:
    apiName: "ata-api"
    replicaCount: 3
    java:
      user_mem_args: "-Xms2000m -Xmx2000m -XX:+HeapDumpOnOutOfMemoryError -
XX:HeapDumpPath=/logMount/$(APP_PREFIX)/ata/ata-api/"
      gc_mem_args: "-XX:+UseG1GC"
      options:
    resources:
      limits:
        cpu: "2"
        memory: 3Gi
      requests:
        cpu: 2000m
        memory: 3Gi
```

Scaling Up or Scaling Down the ATA Service

Provide replica count in **<shape>/ata.yaml** to scale up or scale down the ATA pods. Replica count can be configured for ata-api, topology-consumer, alarm consumer, pgx, and ata-ui pods individually by updating **ata.yaml**.

Update **<shape>/ata.yaml** to increase replica count to 3 for ata-api deployment.

```
ata:
  topologyApi:
    replicaCount: 3
```

Apply the change in replica count to the running Helm release by running the upgrade-applications script.

```
$COMMON_CNTK/scripts/upgrade-applications.sh -p sr -i quick -s $SPEC_PATH -a
ata
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

Enabling GC Logs for ATA

By default, GC logs are disabled, you can enable them and view the logs at the corresponding folders inside location **/logMount/sr-quick/ata**.

To Enable GC logs, update **$SPEC_PATH/sr/quick/applications-base.yaml** file as follows:

1. Under `gcLogs` make `enabled` as `true` you can uncomment `gcLogs` options under ata to override the common values.
2. To configure the maximum size of each file and limit for number of files you need to set `fileSize` and `noOfFiles` inside `gcLogs` as follows:

```
gcLogs:
  enabled: true
  fileSize: 10M
  noOfFiles: 10
```

Debugging and Troubleshooting

Common Problems and Solutions

- ATA DBInstaller pod is not able to pull the dbinstaller image.

```
NAME                                                  READY   STATUS
RESTARTS   AGE
project-instance-unifed-topology-dbinstaller   0/1     ErrImagePull
0          5s

### OR

NAME                                                  READY   STATUS
RESTARTS   AGE
project-instance-unifed-topology-dbinstaller   0/1     ImagePullBackOff
0          45s
```

  To resolve this issue

  1. Verify that the image name and the tag provided in **database.yaml** for ata-dbinstaller and that it is accessible from the repository by the pod.
  2. Verify that the image is copied to all worker nodes.
  3. If pulling image from a repository, verify the image pull policy and image pull secret in **database.yaml** for ata-dbinstaller.

- ATA API, PGX and UI pod is not able to pull the images.

  To resolve this issue

  1. Verify that the image names and the tags are provided in **app-ata.yaml** for ata and that it is accessible from the repository by the pod.
  2. Verify that the image is copied to all worker nodes
  3. If pulling image from a repository, verify the image pull policy and image pull secret in **applications-base.yaml** for ATA service.

Copyright © 2023, 2026, Oracle and/or its affiliates.

- ATA pods are in crashloopbackoff state.

  To resolve this issue, describe the Kubernetes pod and find the cause for the issue. It could be because of missing secrets.

- ATA API pod did not come up.

```
NAME                                          READY   STATUS
RESTARTS   AGE
project-instance-ata-api  0/1     Running   0          5s
```

  To resolve this issue, verify that the **MessagingBusConfig** value provided in **applications-base.yaml** is the valid one.

Test Connection to PGX server

To troubleshoot PGX service, connect to pgx service using graph client by running the following command.

Add **/etc/hosts** file entry for pgx server hostname and loabalancer IP as follows:

```
<loadBalancerIP> <instance>.<project>.topology.<hostSuffix>
```

Connect to pgx service endpoint http://<instance>.<project>.topology.<hostSuffix>/pgx by providing pgx client user credentials.

```
C:\TopologyService\oracle-graph-client-22.1.0\oracle-graph-
client-22.1.0\bin>opg4j -b http://quick.sr.topology.uim.org:30505/pgx -u
<PGX_CLIENT_USER>

password:<PGX_CLIENT_PASSWORD>
For an introduction type: /help intro
Oracle Graph Server Shell 22.1.0
Variables instance, session, and analyst ready to use.
```

Fallout Events Resolution

The fallout events resolution process starts from analyzing the events (or messages) from the **FALLOUT_EVENTS** table. The main intent of this fallout events resolution is to make the message consumer client (such as topology) data in synchronization with the producer client (such as UIM) data by correcting the fallout events which are failed in processing by the consumer clients. Correcting the failed events means rebuilding, resubmitting, editing or ignoring.

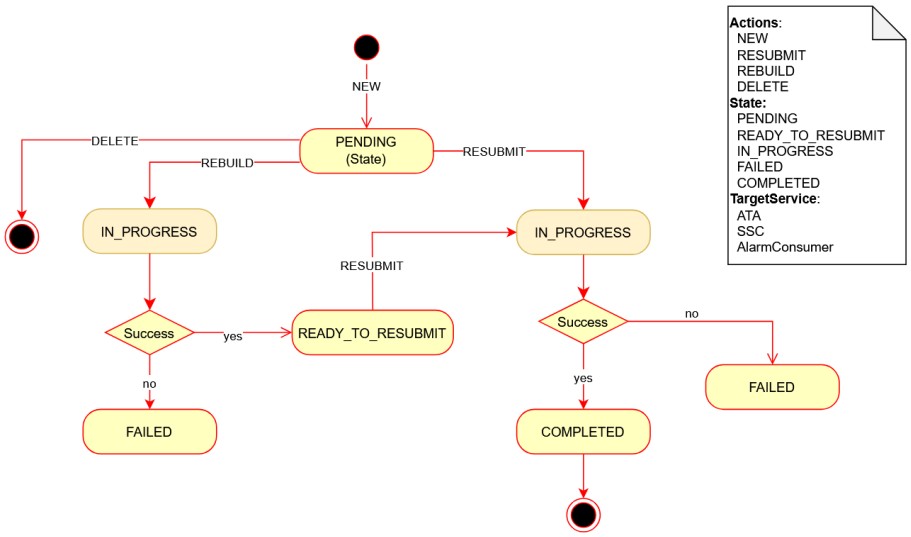
> **Note**
>
> This resolution is a manual administration task that should be performed periodically either using the Message Reconciliation User Interface or Fallout Events Resolution RESTful endpoints.

At first, these events (or messages) are collected in the **FALLOUT_EVENTS** table. The message consumer services subscribe and consume the events from the message bus on a specific topic (such as **ora-uim-topology)** and process them accordingly. Process involves

Copyright © 2023, 2026, Oracle and/or its affiliates.

extracting the event type from the message and create, update, or delete the entity from the corresponding downstream entities for each event. For any failures occurred, the system retries for the configured number of times. If there are any failures that are still being processed, the corresponding messages are sent to the consumer service-specific **dead-letter**-topic (such as **ora-dlt-topology)**. A dedicated subscriber then consumes from this **dead-letter** topic, inserts these events (or messages) into the **FALLOUT_EVENTS** table for further handling.

The general fallout events resolution process flow is illustrated in the following image. Some of process steps are applicable only to specific message consumer services. For example, the REBUILD setup is applicable only to the topology consumer service.



The new Message Reconciliation application helps you in resolving the fallout events. Oracle recommends to use this Message Reconciliation user interface for resolving your fallout events and also encourages to use the fallout resolution end-points for any headless automation. See *Active Topology Automator and Service Impact Analysis User's Guide* for details on the Message Reconciliation UX flow.

The following actions can be performed on fallout events. The actions can be performed either from the message reconciliation UX or using the REST APIs provided as part of ATA-API.

Table 8-1 Fallout Events Actions

Action	Description
NEW	The event is added to the **FALLOUT_EVENTS** table if the action is **NEW**.
REBUILD	This action corrects the missing dependencies. **Note**: This action is applicable only to the topology consumer. The Kafka header **rebuild_count** tracks the number of times the rebuild action is triggered for an event.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Table 8-1 (Cont.) Fallout Events Actions

Action	Description
RESUBMIT	The resubmit action resubmits the event or message into the retry topic (such as **ora-retry-topology**) of the specific target service for reprocessing. The Kafka header **resubmit_count** tracks the number of times the resubmit action is triggered for an event. The Kafka header **previous_fallout_eid** identifies the previous fallout event from which the current fallout was created.
DELETE	Clears the event or messages from the fallout tables.
EDIT	Edits the specific event or message from the fallout table.

The following are various states that an event can be based on the corresponding actions.

Table 8-2 Fallout Events States

State	Description
PENDING	The fallout event is in pending state for the newly arrived fallouts.
IN_PROGRESS	The fallout event is in progress for either RESUBMIT or REBUILD actions.
READY_TO_RESUBMIT	The fallout event is processed by the REBUILD action and is ready for RESUBMIT.
FAILED	The fallout event processing is failed for either RESUBMIT or REBUILD.
COMPLETED	The fallout event processing is completed for RESUBMIT.

You can use the fallout REST APIs for resolving these failures. See *REST API for ATA* for full list of REST APIs.

Table 8-3 REST APIs and the Corresponding Resolutions

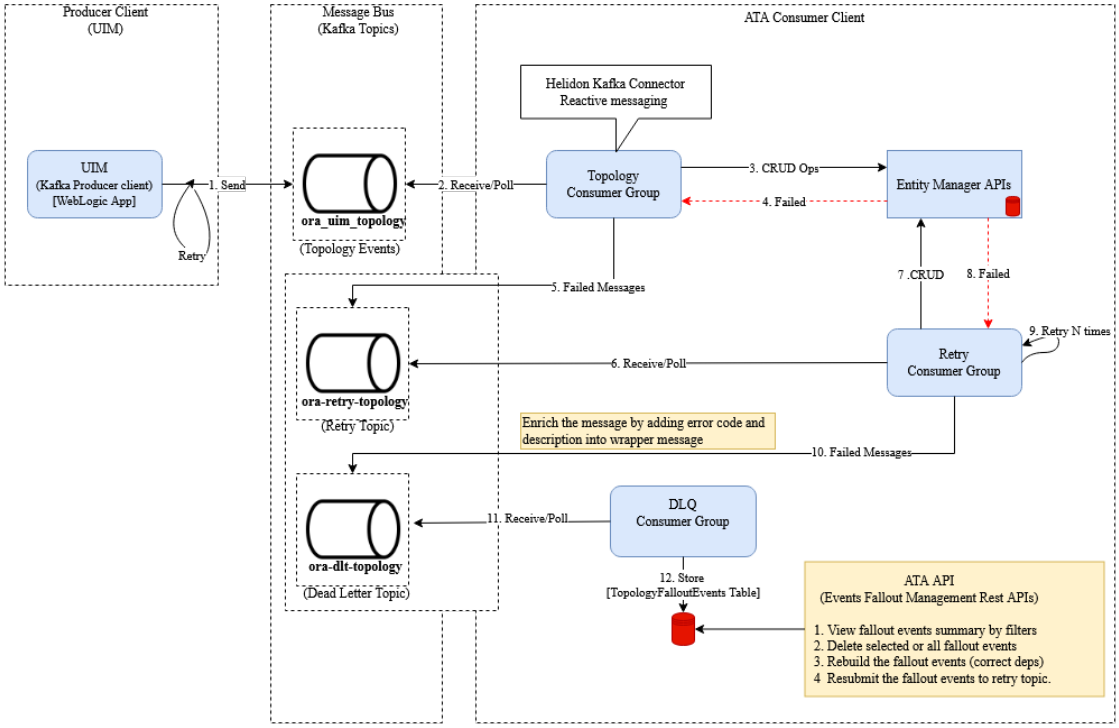
REST API	Resolution
/topology/v2/fallout/events/summary	Get a brief summary of the fallout events with status.
/topology/v2/fallout/events/rebuild/jobs	Rebuild to correct the missing dependencies.
/topology/v2/fallout/events/resubmit	Resubmit to retry the topic to re-process the event or message.
/topology/v2/fallout/events /topology/v2/fallout/events/eid/{eid}	Clear or delete the event or message from the fallout table.
/topology/v2/fallout/events/eid/{eid}	To edit the event or message in the fallout table and resubmit again.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Fallout Events Resolution for Topology Consumer

The following figure illustrates the fallout events resolution process flow for topology consumer.

Figure 8-1 Process Flow of Fallout Events Resolution for Topology Consumer



These failed events in the TOPOLOGY_FALLOUT_EVENTS table can be rebuilt and resubmitted for the specific target service identifier. The target service value used for the topology consumer is **ATA**. When a fallout event comes into the table, it is in **PENDING** state. These events can be Rebuilt or Resubmitted as follows:

- REBUILD: This action processes the Fallout Event and gets any out of sync data from UIM into ATA through the Database Link.
- RESUBMIT: This action takes the events from the TOPOLOGY_FALLOUT_EVENTS table in **PENDING** or **READY_TO_RESUBMIT** states and moves them back into the **ora_retry_topology** topic for re-processing.

Prerequisites for REBUILD

- Before Rebuild is performed, the ATA Schema user should have the following privileges:
  - CREATE JOB
  - ALTER SYSTEM
  - CREATE DATABASE LINK
- Ensure a Database Link exists from ATA schema to UIM schema with the name **REM_SCHEMA**. That is, ATA schema user should be able to access the objects from UIM schema. For more information, see https://docs.oracle.com/en/database/oracle/oracle-

Copyright © 2023, 2026, Oracle and/or its affiliates.

database/19/sqlrf/CREATE-DATABASE-LINK.html#GUID-D966642A-B19E-449D-9968-1121AF06D793

Performing REBUILD Action

You can perform the Rebuild action in the following ways:

- **DBMS Job Scheduling:** In this approach the REBUILD action on the Fallout Events in “PENDING” state is scheduled to run for every 6 hours. The frequency at which the job runs automatically can be configured by changing the **repeat_interval**.

```
BEGIN
  DBMS_SCHEDULER.create_job (
    job_name        => 'FALLOUT_DATA_REBUILD',
    job_type        => 'PLSQL_BLOCK',
    job_action      => 'BEGIN
PKG_FALLOUT_CORRECTION.SCHEDULE_FALLOUT_JOBS(commitSize => 1000, cpusJobs
=> 4, waitTime => 2); END;',
    start_date      => SYSTIMESTAMP,
    repeat_interval => 'FREQ=HOURLY; INTERVAL=6',
    enabled         => TRUE
  );
END;
/
```

- **On-Demand REST API Call:** In this approach, the REBUILD action on the Fallout Events in **PENDING** state are invoked through REST API before invoking the Rebuild API:
  - POST - fallout/events/rebuild – To rebuild the Fallout Events on demand as and whenever required.
  - DELETE - fallout/events/scheduledJobs – To drop any running or previously scheduled jobs.

Performing RESUBMIT Action

Resubmit Action is performed through a REST call and it takes the fallout events in “READY_TO_RESUBMIT” (post Rebuild) and **PENDING** states based on the query parameters and pushed the events into the “ora_retry_topology” topic:

POST - fallout/events/resubmit?targetService=ATA – To resubmit the Fallout Events on demand for topology consumer.

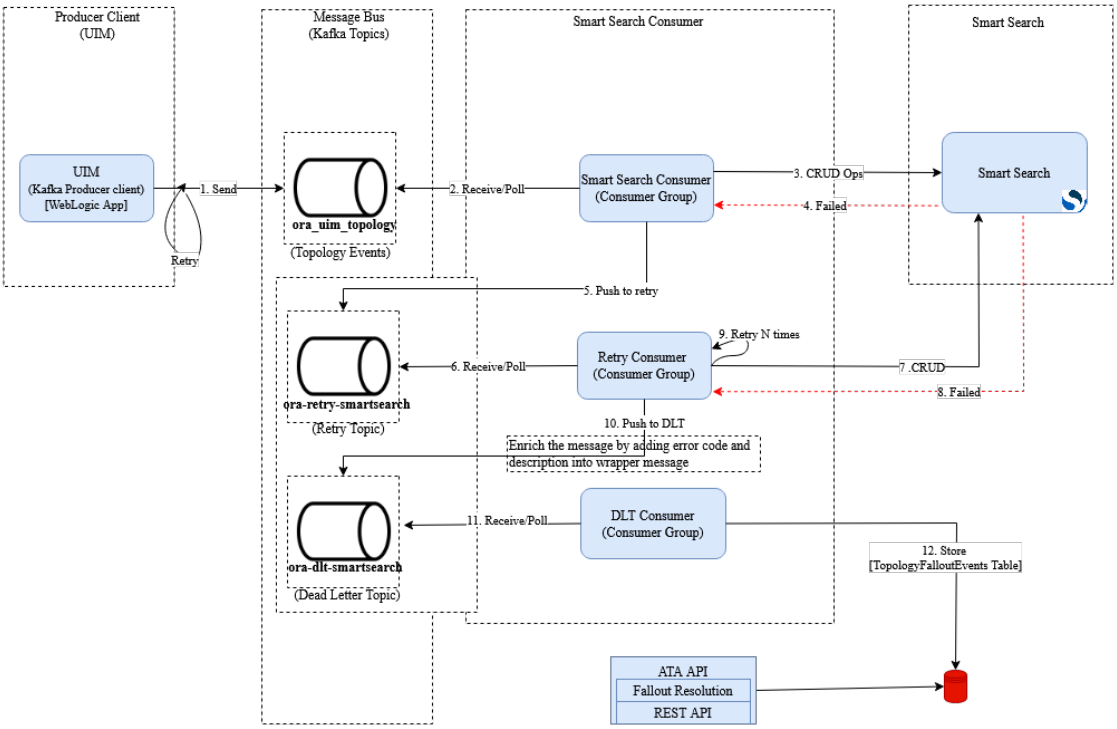
For more information on APIs available, see *ATA REST API Guide*.

Fallout Events Resolution for SmartSearch Consumer

The events will be stored into topology fallout event table when the SmartSearch consumer service is not able to process successfully due to some possible technical and data validation issues.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Figure 8-2 Process Flow of Fallout Events Resolution for SmartSearch Consumer



The fallout resolution for a SmartSearch consumer is as follows:

1. The message producer sends event and message to the topic (**ora-uim-topology**).
2. The SmartSearch consumer receives the event and message from the topic (**ora-uim-topology**).
3. It invokes the SmartSearch bulk API to persist the event or message in Open Search.
4. If the SmartSearch bulk API not able to process the even or message, or SmartSearch consumer is not able to process then the SmartSearch consumer sends a message to retry the topic (**ora-retry-smartsearch)**.
5. The SmartSearch consumer retries again to process. If still not processed, then the SmartSearch consumer sends an event message to the Dead Letter topic (**ora-dlt-smartsearch**).
6. The DLT process of SmartSearch consumer gets event or message from the Dead Letter topics that persist in the **TOPOLOGY_FALLOUT_EVENTS** table.

The fallout resolution starts analyzing from the topology fallout events for the specific target service. The target service value used for SmartSearch consumer is **SSC**. The resolution can be performed with the help of fallout resolution REST APIs.

Table 8-4 Fallout Events REST APIs

REST API	Resolution
/topology/v2/fallout/events/events/summary?targetService=SSC	Get the fallout events summary for the SmartSearch consumer.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Table 8-4 (Cont.) Fallout Events REST APIs

REST API	Resolution
/topology/v2/fallout/events/events?targetService=SSC	Get all events by matching the state and action for the SmartSearch consumer. Query Parameters: • **state:** PENDING/FAILED • **action:** NEW/RESUBMIT • **targetService:** SSC
/topology/v2/fallout/events/resubmit	Re-submit the matched events of SmartSearch consumer for re-processing Query Parameters: • **state:** PENDING/FAILED • **action:** NEW/RESUBMIT • **targetService:** SSC
/topology/v2/fallout/events/eid/{ENTITY_ID}	Update a specific fallout.
/topology/v2/fallout/events /topology/v2/fallout/events/eid/{ENTITY_ID}	To delete matching the fallout events for SmartSearch consumer. The parameter values are as follows: • **state:** PENDING/FAILED • **action:** NEW/RESUBMIT • **targetService:** SSC

For full list of REST APIs, see *REST API for ATA for Inventory and Automation*.

ATA Support for Offline Maps

ATA support for map visualization is provided by the third-party service providers such as Open Street Maps (OSM), MapBox, Carto, Esri, and Web Map Service (WMS).

ATA integrates with these service providers and they provide the required components and computing resources, so that you can avoid setting up and maintaining a local tile server.

Oracle offers the following options to support offline maps:

- Allowlisting map URLs
- Setting up a local tile server

Allowlisting Map URLs

In highly secured installations, you may not provide internet access to the location. In such situations, Oracle recommends using an allowlist solution so the base maps can include the streets, cities, buildings, and so on.

For the map tiles to render, allowlist the following URLs:

- Tile 1:
  - http://a.tile.openstreetmap.org/11/472/824.png
  - http://b.tile.openstreetmap.org/11/472/825.png
  - http://c.tile.openstreetmap.org/11/472/825.png
- Tile 2:

Copyright © 2023, 2026, Oracle and/or its affiliates.

  - http://a.tiles.mapbox.com/v4/mapbox.satellite/10/236/412@2x.png?access_token=pk.eyJ1IjoidzhyIiwiYSI6ImNpeGhwaXF1ejAwMHQydG8yZ3pyanZ5aTkifQ.QNScWNGnLRHIHXeAsGMvyw
  - http://b.tiles.mapbox.com/v4/mapbox.satellite/10/235/411@2x.png?access_token=pk.eyJ1IjoidzhyIiwiYSI6ImNpeGhwaXF1ejAwMHQydG8yZ3pyanZ5aTkifQ.QNScWNGnLRHIHXeAsGMvyw
  - http://c.tiles.mapbox.com/v4/mapbox.satellite/10/236/411@2x.png?access_token=pk.eyJ1IjoidzhyIiwiYSI6ImNpeGhwaXF1ejAwMHQydG8yZ3pyanZ5aTkifQ.QNScWNGnLRHIHXeAsGMvyw
- Tile 3:
  - http://a.basemaps.cartocdn.com/light_all/10/235/412@2x.png
  - http://b.basemaps.cartocdn.com/light_all/10/235/412@2x.png
  - http://c.basemaps.cartocdn.com/light_all/10/235/412@2x.png
- Tile 4:
  - http://a.basemaps.cartocdn.com/dark_all/10/236/413@2x.png
  - http://b.basemaps.cartocdn.com/dark_all/10/236/413@2x.png
  - http://c.basemaps.cartocdn.com/dark_all/10/236/413@2x.png
- Tile 5: http://server.arcgisonline.com/ArcGIS/rest/services/World_Street_Map/MapServer/tile/10/412/235
- Tile 6: http://server.arcgisonline.com/ArcGIS/rest/services/World_Imagery/MapServer/tile/10/412/236

Setting Up a Local Tile Server

To set up a local tile server, you must deploy the prepared tile files on a local server. Every file must have its direct link: **http://tileserver.com/{z}/{y}/{x}**. This format allows getting the required response for the request **http://tileserver.com/{z}/{x}/{y}.png**.

Contact your system administrator to install, deploy, and run your own tile server. The configuration process is dependent on the tile server you choose to implement. The tile server requires high computing power and requires operations support and maintenance.

The tile server is responsible for caching the tiles, sharing the load, and processing the request queue at regular intervals.

You can consider some options available in the market such as MapTiler, QGIS, Switch2OSM, ArcGis Enterprise, and so on.

After you set up the tile server and a successful deployment, you can access the map tiles through APIs in the format: **http://{hostname}:{port}/{baseUrl}/{z}/{x}/{y}.png**.

Manual Changes for Setting Up a Local Tile Server

The following manual changes are required to set up a local tile server:

- Update **visualization-start-page.js** as per your requirement.
- Open **ata-ui.jar** and navigate to **ata-ui/flows/visualization/pages/visualization-start-page.js**.
- In the **loadGeoMaps** method of **visualization-start-page.js**, update the **mapurl** variable of the custom map API URL.

Copyright © 2023, 2026, Oracle and/or its affiliates.

After you redeploy the updated jar file and run the application, you can see the map tiles served from your local server.

Copyright © 2023, 2026, Oracle and/or its affiliates.

9
Upgrading ATA

This chapter describes how to upgrade the ATA application.

Prerequisites for Upgrading ATA

The prerequisites for upgrading ATA are:

- ATA Schema should have a database link to the UIM schema with the name `rem_schema`. This is mandatory if only ATA is used with UIM. However, the database link is not required if ATA is used with some external system. The `rem_schema` database link is created during the first time of complete migration. If the database link is not present, the database link can be created as follows:

```
ACCEPT schema CHAR PROMPT "Enter username for remote schema: "
ACCEPT passwd CHAR PROMPT "Enter password for remote schema: " HIDE
ACCEPT host CHAR PROMPT "Enter pingable hostname/ipaddress for remote
schema database host : "
ACCEPT port CHAR PROMPT "Enter port number for remote schema database : "
ACCEPT service_name CHAR PROMPT "Enter SQL*Net / service for remote schema
database: "
ACCEPT commitSize CHAR PROMPT "Enter Batch/Commit size for a single
parallel process(Optional): "
ACCEPT threads CHAR PROMPT "Enter Maximum no.of total parallel process at
any given time(Optional): "
ACCEPT waitTime CHAR PROMPT "Enter Waiting interval after which the
listener checks for the availabilty of jobs in Seconds(Optional): "

PROMPT

alter system set global_names=FALSE scope=both;

CREATE DATABASE LINK rem_schema CONNECT TO &schema IDENTIFIED BY &passwd
USING '(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=&host)(PORT=&port))
(CONNECT_DATA=(SERVICE_NAME=&service_name)))';
```

- For ATA 1.0.0.1.0 or later versions, the installer will create an **ApplicationInfo** table and will update the **VERSION** after every upgrade. If you have ATA 1.0.0.0.0 installed, you will not be having **ApplicationInfo** table. Therefore, create **ApplicationInfo** table before running an upgrade as follows:

```
CREATE TABLE APPLICATIONINFO ( ENTITYID NUMBER(19,0) NOT NULL ENABLE,
  ENTITYCLASS VARCHAR2(255 BYTE),
  BUILDDATE TIMESTAMP (6) WITH LOCAL TIME ZONE,
  CREATEDDATE TIMESTAMP (6) WITH LOCAL TIME ZONE,
  CREATEDUSER VARCHAR2(255 BYTE),
  ENDDATE TIMESTAMP (6) WITH LOCAL TIME ZONE,
  ENTITYVERSION NUMBER(10,0),
  FILENAME VARCHAR2(255 BYTE),
  LASTMODIFIEDDATE TIMESTAMP (6) WITH LOCAL TIME ZONE,
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
  LASTMODIFIEDUSER VARCHAR2(255 BYTE),
  NAME VARCHAR2(255 BYTE),
  STARTDATE TIMESTAMP (6) WITH LOCAL TIME ZONE,
  STATUS VARCHAR2(255 BYTE),
  TYPE VARCHAR2(255 BYTE),
  VERSION VARCHAR2(255 BYTE),
   PRIMARY KEY (ENTITYID));

   INSERT INTO APPLICATIONINFO VALUES (ENTITYID_SEQ.NEXTVAL,
'ApplicationInformationDAO', SYSDATE, SYSDATE, NULL, SYSDATE, 1, NULL,
SYSDATE, NULL, 'Active Topology Automation', SYSDATE, 'SUCCESS',
'Topolgy', '1.0.0.0.0');
```

Upgrading the ATA Application

To upgrade the ATA application:

1. Download the latest ATA Builder Tool Kit and Common Cloud Native Tool Kit into the workspace directory.
2. Export the unzipped path to the **WORKSPACEDIR** environment variable.

   ```
   export WORKSPACEDIR=$(pwd)/workspace
   ```

3. Set the **COMMON_CNTK** variable to the path of the common-cntk directory in the workspace.

   ```
   export COMMON_CNTK=$WORKSPACEDIR/common-cntk
   ```

4. Set **SPEC_PATH** variable to the location where **applications-base.yaml**, **app-ata.yaml**, **<shape>/ata.yaml**, and **database.yaml** files are assembled :

   ```
   $ export SPEC_PATH=$WORKSPACEDIR/spec_dir
   ```

5. Create ATA images using the latest ATA Builder Tool Kit. See "Prerequisites and Configuration for Creating ATA Images" for more information.
6. Upgrade the ATA schema. See "Upgrading the ATA Schema" for more information.
7. Upgrade the ATA instance. See "Upgrading the ATA Instance" for more information.

Upgrading the ATA Schema

To upgrade the ATA schema:

1. Upgrade PDB by starting $UIM_CNTK/scripts/install-database.sh.
2. To only update the model of ATA and skip the data migration:

   ```
   $COMMON_CNTK/scripts/install-database.sh -p sr -i quick -s $SPEC_PATH -a
   ata -c 4
   ```


Copyright © 2023, 2026, Oracle and/or its affiliates.

3. To update the model of ATA and also populate the data from the UIM schema:

```
$COMMON_CNTK/scripts/install-database.sh -p sr -i quick -s $SPEC_PATH -a
ata -c 40
```

4. If there are any changes to Ingress configurations, loadbalancer port, and annotations, and so on, delete and create ATA Ingress as follows:

```
$COMMON_CNTK/scripts/delete-ingress.sh -p sr -i quick -s $SPEC_PATH -a ata

$COMMON_CNTK/scripts/create-ingress.sh -p sr -i quick -s $SPEC_PATH -a ata
```

Upgrading the ATA Instance

To upgrade the ATA instance:

1. Update $COMMON_CNTK/samples/applications.yaml with the latest ATA API, ATA PGX, and ATA UI image names and the corresponding tags.
2. Run $COMMON_CNTK/scripts/upgrade-applications.sh to upgrade the ATA instance:

```
$COMMON_CNTK/scripts/upgrade-applications.sh -p sr -i quick -s $SPEC_PATH -
a ata
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

10

Localization

This chapter provides information on localizing the Active Topology Automation (ATA), Service Impact Analysis, and Message Reconciliaiton user interface (UI). The localization is a process of translating a UI from the original language in which it was written into a different language for use in a specific country or region.

The supported languages for localization are: **fr-ca**, **fr**, **es**, **en**.

The localization process works as follows:

- If a regional variant of one of the above languages is selected but is not available, the system rolls back to its appropriate parent language (for example: es-ar → es, en-in → en). A message appears as follows:

  ```
  The page is displayed in <parentLanguage> because <parentLanguage-
  regionalVariant> is not supported.
  ```

- If a different language is selected that is not one of the above mentioned languages and has no parent language included among them, the system rolls back to English (en). A message appears as follows:

  ```
  The page is displayed in English because <selected-language> is not
  supported.
  ```

Localizing the UI involves modifying a specific set of files to display text in the UI. It requires the following bundles:

- The **ATA app** bundle
- The **Message Reconciliation app** bundle
- The **UIM app** bundle

Localizing the API requires the properties files in ATA API.

About App Bundles

The **app** bundles are a set of **.js** files that contain text strings that can be localized, that define labels and messages in UI. You can find the app bundles at the following location within **ata-ui.jar**: **$BUILDER_HOME/staging/downloads/ata-ui**. The **/root/appBundle-strings.js** is the default app bundle and it contains the strings in English language.

Localizing the ATA App Bundles

To localize the ATA app bundles:

1. Add a folder at **$BUILDER_HOME/staging/downloads/ata-ui/ataAppBundle/nls/** with locale code as the name of the folder.
2. Copy **$BUILDER_HOME/staging/downloads/ata-ui/ataAppBundle/nls/root/appBundle-strings.js** to the newly created folder.

Copyright © 2023, 2026, Oracle and/or its affiliates.

3. The **appBundle-strings.js** contains the key-value pairs, where value is the label or message displayed in the UI.
4. Edit the values in the copied **appBundle-strings.js** file with the required localized strings.
5. In **$BUILDER_HOME/staging/downloads/ata-ui/ataAppBundle/nls/appBundle-strings.js**, add the locale code as label and **true** as the value. You can use the following sample:

```
define({
    "root": true,
    "it": true,
    "fr": true,
    "sv": true
});
```

The sample app bundles for default root bundle (en) and the fr bundle are as follows:

- **root/appBundle-strings.js**

```
define({
    "outofText" : "out of",
    "@outofText": {
      "description": ""
    },
    "uploaded": "uploaded",
    "@uploaded": {
      "description": ""
    },
    "all": "All",
    "@all": {
      "description": ""
    }
 })
```

- **fr/appBundle-strings.js**

```
define({
    "outofText": "hors de",
    "@outofText": {
        "description": ""
    },
    "uploaded": "téléchargé",
    "@uploaded": {
        "description": ""
    },
    "all": "All",
    "@all": {
        "description": ""
    }
})
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

> **Note**
>
> Modify the labels (key) values as shown in the samples and not in the **appBundle-strings.js**.

Localizing the Message Reconciliation App Bundles

To localize the Message Reconciliation UX app bundles:

1. Create a new folder at **$BUILDER_HOME/staging/downloads/ata-ui/msgReconAppBundle/nls/** with the locale code as the folder name.
2. Copy the **msgReconAppBundle-strings.js** file to the newly created folder.
3. The **msgReconAppBundle-strings.js** file contains key-value pairs, where the value represents the label or message displayed in the corresponding UI. Modify the values in the copied **msgReconAppBundle-strings.js** file with the appropriate localized strings for the specific locale.
4. In the **$BUILDER_HOME/staging/downloads/ata-ui/msgReconAppBundle/nls/msgReconAppBundle-strings.js** file, add an entry for the locale code with the value as `true` as follows:

```
define({

          "root": true,

           "fr": true

});
```

The samples of the app bundles for the default root bundle (en) and the French (fr) bundle are as follows:

root/msgReconAppBundle-strings.js

```
define({

      "OVERVIEW_HEADER": "Overview",

      "@OVERVIEW_HEADER": {

             "description": "Overview Menu Header"

       },

      "SELECT_CONSUMER": "Select Consumer",

      "@SELECT_CONSUMER": {

            "description": "Select Consumer Label"

      }

})
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

fr/msgReconAppBundle-strings.js

```
define({

      "OVERVIEW_HEADER": "Aperçu",

      "@OVERVIEW_HEADER": {

              "description": "Overview Menu Header"

       },

      "SELECT_CONSUMER": "Sélectionner un consommateur",

      "@SELECT_CONSUMER": {

             "description": "Select Consumer Label"

      }

})
```

> **Note**
>
> Modify the labels (key) values as shown in the samples and not in the **msgReconAppBundle-strings.js** file.

About Properties in ATA API

ATA API generates messages in the API response that are displayed in ATA UI in certain flows. These messages are a part of the properties files and located at **$BUILDER_HOME/staging/downloads/ata-api/logging/resources/ui.properties**. The format for **ui.properties** file is as follows:

```
ui.{key}.id={id}
ui.{key}={value or message to be displayed}
```

Localizing Properties in ATA API

To localize the properties in ATA API:

1. Create a file using the following naming convention at **$BUILDER_HOME/staging/downloads/ata-api/logging/resources/**:

   ```
   ui_{locale_code}.properties
   # example: ui_fr.properties for french
   ```

2. Copy the contents of **ui.properties** to **ui_*<locale>*.properties**.
3. Modify the values in **ui_*<locale>*.properties** to the required locale (For example: **fr**).
4. Build the ATA API application and run to update the application with the locale changes.

Copyright © 2023, 2026, Oracle and/or its affiliates.

The sample **ui.properties** and **ui_fr.properties** properties are as follows:

- **ui.properties**

```
ui.invalidSearch.id=5000
ui.invalidSearch=A valid auto suggest search term is required : {0}

ui.query.id=5001
ui.query=Following query is prepared to run : {0}

ui.invalidSearchName.id=5002
ui.invalidSearchName=Search Name is required.
```

- **ui_fr.properties**

```
ui.invalidSearch.id=5000
ui.invalidSearch=Un terme de recherche de suggestion automatique valide
est requis : {0}

ui.query.id=5001
ui.query=La requête suivante est prête à être exécutée : {0}

ui.invalidSearchName.id=5002
ui.invalidSearchName=Le nom de recherche est obligatoire.
```

> **Note**
>
> Modify the values as shown in the samples and not in the id or keys.

Along with the above properties, modify the following properties file for complete API localization:

- vertext.properties
- edge.properties
- graph.properties
- tmfresp.properties
- falloutEvent.properties.

About UIM App Bundle in ATA

You can localize the specification names in ATA UI by exporting the specification bundle from UIM and building the image with customization.

Before you localize the specification name, make sure that the specification display names are provided in Design Studio and deployed to UIM.

Localizing Specification Names in ATA UI

To localize the specification names in ATA UI:

1. Export the **UIM App** bundle:

Copyright © 2023, 2026, Oracle and/or its affiliates.

a. In the UIM left navigation pane, go to **Execute Rule** under the **Administration** section.

b. Select **EXPORT_SPECIFICATION_DISPLAY_NAMES_AS_JSON** from the dropdown, ignore the file upload option, and click **Process**.

> **Note**
>
> For this option to appear, the **ora_uim_baserulesets** should be deployed.

c. Download **uimAppBundle.tar.gz**.

2. Add the builder for image building by adding **uimAppBundle.tar.gz** in **$BUILDER_HOME/staging/downloads/ata-ui/uimAppBundle/**.

3. Customize and build the image.

Copyright © 2023, 2026, Oracle and/or its affiliates.

11

Deploying Service Impact Analysis

This chapter describes how to deploy and manage Service Impact Analysis.

Service Impact Analysis Overview

Service Impact Analysis enables you to view the alarm events associated with UIM resources and view the impacts to customer, service, network, logical, and physical resources, and connectivity. It also enables you to assign ownership to specific individuals and track the impact lifecycle using the analysis process. These alarm events are associated to the corresponding UIM resource through the Alarm Consumer service, using the *ora-alarm-topology* topic with TMF642 alarm event JSON format.

For the architecture, see "ATA Architecture".

You must deploy ATA before deploying Service Impact Analysis.

Creating Service Impact Analysis Images

You must have created the Service Impact Analysis images as part of "Prerequisites and Configuration for Creating ATA Images" and "Creating ATA Images".

Verify if the following images are available:

- uim-<uim-version>-alarm-consumer-<ata-version>:latest
- uim-<uim-version>-impact-analysis-api-<ata-version>:latest

Creating Service Impact Analysis Instance

The Service Impact Analysis instance is dependent on the ATA Instance to be deployed.

Prerequisites:

- Deploy ATA using "Creating an ATA Instance".
- Create the required secrets and other configurations (such as Authentication, Oracle Database schema, SmartSearch) while deploying ATA.

Configuring the Application Specifications Files

To configure the **app-ata.yaml** file:

1. Edit the **app-ata.yaml** file to provide the image in your repository (name and tag) by running the following command:

   ```
   vi $SPEC_PATH/$PROJECT/$INSTANCE/app-ata.yaml
   ```


Copyright © 2023, 2026, Oracle and/or its affiliates.

2. Edit the image names to reflect the Service Impact Analysis image names and location in your docker repository as follows:

```
ata:
  name: "ata"
  image:
    alarmConsumerName: uim-<uim-version>-alarm-consumer-<ata-version>
    impactAnalysisApiName: uim-<uim-version>-impact-analysis-api-<ata-
version>
    alarmConsumerTag: latest
    impactAnalysisApiTag: latest
    repository:
    repositoryPath:
```

3. Edit the **$SPEC_PATH/$PROJECT/$INSTANCE/shapes/<shape>/ata.yaml** file to update the `replicaCount` of **alarmConsumer** and **impactAnalysisApi**. The sample configuration is as follows. Update the replica count according to your performance needs:

```
alarmConsumer:
  name: "alarm-consumer"
  replicaCount: 3

impactAnalysisApi:
  name: "impact-analysis-api"
  replicaCount: 3
```

Configuring Service Impact Analysis

This section helps you to configure Service Impact Analysis.

Configuring UIM

Impact correlation for the events submitted through alarm consumer in the system is done at the UIM side, which then can be viewed on the Service Impact Analysis UI. See "About SIA" for more information.

Configuring Service Impact Analysis API

Service Impact Analysis provides APIs for persisting and managing events and impact correlation of these events. It uses OpenSearch indexes as its persistence mechanism.

Sample configuration files **impactanalysis-static-config.yaml.sample** and **impactanalysis-dynamic-config.yaml.sample** are provided as the sample files for Impact Analysis API service that are under **$SPEC_PATH/$PROJECT/$INSTANCE/config/ata/impact-analysis-api**.

To override configuration properties, copy the sample static property file to **impactanalysis-static-config.yaml** and sample dynamic property file to **impactanalysis-dynamic-config.yaml**. Provide key value to override the default value provided for any specific system configuration property. The properties defined in property files are entered in the container using Kubernetes configuration maps. Any changes to these properties require the instance to be upgraded. Restart the pods after updating the configuration changes to **impactanalysis-static-config.yaml**.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Date Format

Any modifications to the date format used by all dates must be consistently applied to all consumers of the APIs. API serializes and deserializes the date attributes stored in OpenSearch indexes using following date format:

```
impactanalysis:
  api:
    dateformat: yyyy-MM-dd'T'HH:mm:ss.SSS'Z'
```

Event Status

Service Impact Analysis API supports the following types of events:

- **RAISED**: This event type is for new events.
- **UPDATED**: This event type is for existing events with updated information.
- **CLEARED**: This event type is for events that have been Closed.
- **REJECTED**: This event type is for events that are invalid and are rejected through **Reject Event** action available on Service Impact Analysis UI.

> **Note**
>
> Rejected events move to a different index called `smartsearch-rejected-event` and are not deleted. Rejecting events is to isolate invalid events that are submitted to the system.

The following event statuses, apart from **REJECTED** are standard TMF642 event statuses. These event status mappings are part of Unified Assurance integration and should not be changed:

```
impactanalysis:
  event-status:
    CLEARED: CLEARED
    RAISED: RAISED
    UPDATED: UPDATED
    REJECTED: REJECTED
```

Event Severity

Service Impact Analysis API and ATA support various types of event severities on a Device. The severities from most severe to least severe are CRITICAL (1), MAJOR (5), WARNING (10), INTERMEDIATE (15), MINOR (20), CLEARED (25), and None (999). Internally, a numeric value is used to identify the severity hierarchy. The top three most severe events (CRITICAL, MAJOR, WARNING) are tracked in ATA.

The following event severities are standard TMF642 event severities. These event severity mappings are part of Unified Assurance integration and should not be changed:

```
impactanalysis:
  severity:
    CLEARED: CLEARED
    INDETERMINATE: INDETERMINATE
    CRITICAL: CRITICAL
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
    MAJOR: MAJOR
    MINOR: MINOR
    WARNING: WARNING
```

Impact Calculation Thread Pool Size

The impact calculation thread pool size defines the maximum number of open REST requests at a time to UIM for fetching impacts.

```
impact:
  threadPoolSize: 10
```

SmartSearch and OpenSearch Related Configurations

The configurations related to SmartSearch and OpenSearch are as follows:

- **smartsearch.lang**: The value of SmartSearch internal field used in lang pipeline processing. The only supported value is `en`.
- **smartsearch.tenantId**: The value of SmartSearch internal field for search tenancy. The only supported value is `tenant1`.
- **smartsearch.fetchSize**: Defines the number of documents that should be fetched at a time in memory from SmartSearch for processing during bulk operations. The maximum limit for this value is `10000`.
- **smartsearch.coolDownInterval**: Defines the cool-down interval in milliseconds for OpenSearch index after each batch of documents is processed (bulk updates and deletes).

```
smartsearch:
  language: en
  tenantId: tenant1
  fetchSize: 1000
  coolDownInterval: 1000
```

Events Related Configurations

The configuration related to events are as follows:

- **event.idPrefix**: Defines the prefix used during document id generation for OpenSearch event index.
- **event.reportPrefix**: Defines the prefix used during report id generation when the analysis status of the event transitions to **COMPLETED**.
- **event.defaultOwner**: Defines the default owner name value to be used in case the owner field is not populated during event creation.

```
event:
  idPrefix: UE
  reportPrefix: REP
  defaultOwner: Unassigned
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

Configuring Alarm Consumer

Sample configuration files **alarm-consumer-static-config.yaml.sample** and **alarm-consumer-dynamic-config.yaml.sample** are provided under **$SPEC_PATH/$PROJECT/$INSTANCE/config/ata/alarm-consumer**.

To override configuration properties, copy the sample static property file to **alarm-consumer-static-config.yaml** and sample dynamic property file to **alarm-consumer-dynamic-config.yaml**. Provide key value to override the default value provided out-of-the-box for any specific system configuration property. The properties defined in property files are provided to the container using Kubernetes configuration maps. Any changes to these properties require the instance to be upgraded. Restart the pods after updating the configuration changes to **alarm-consumer-static-config.yaml**.

The alarm consumer service receives the alarm event notifications in TMF642 v5.0 specification JSON string format (TMF642 alarm JSON wrapped in TMF688 event JSON) from **ora-alarm-topology** Kafka topic. As part of the alarm event notifications processing, alarm is created and associated with the effected entity (node or sub-node) in the Service Impact Analysis service to resolve the use cases. Based on the event type in the notification, the alarms can be updated, cleared, or deleted from the affected entity.

The default implementation in processing an alarm is to retrieve the entity (device and sub-device) from Inventory (UIM or ATA) by filtering with name and entity type and associate the alarm. The **alarmedObject.id** element value, in the alarm object, represents the name of entity (the device and sub-device identification is separated by "::" delimiter). The **alarmObject.@referredType** element value represents entity type (device or sub-device type) and the value for the **@referredType** element represents the TMF639 sub-type for the resource on which the alarm is raised. The resource sub-types are listed in the following sections.

The following sections list the extension available to configure or customize the entity look logic:

- Resource Type mappings
- Customizing Device Mapping
- Alarmed object extension

Following are the samples on the **alarmedObject** sub-structure from the TMF642 alarm event specification. For more details on the full event payload, see the *Active Topology Automation Asynchronous Events Guide*:

Sample alarmedObject sub structure for device alarm

```
{
  "eventId": "700001",
  "@type": "AlarmCreateEvent",
  "eventType": "AlarmCreateEvent",
  "event": {
    "alarm": {
      .....
      "alarmedObject": {
        "@referredType": "PhysicalDevice",
        "@type": "AlarmedObjectRef",
       "id": "LSN/EMS_XDM_33/9489"
      },
     ...
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
   }
 }
```

Sample alarmedObject sub structure for the sub-device (port) alarm

```
{
  "eventId": "700001",
  "@type": "AlarmCreateEvent",
  "eventType": "AlarmCreateEvent",
  "event": {
    "alarm": {
      .....
      "alarmedObject": {
        "@referredType": "PhysicalPort",
        "@type": "AlarmedObjectRef",
       "id": "LSN/EMS_XDM_33/9489::LSN/EMS_XDM_33/P01-142.1K.07-Line-
Card1.OTU4_8"
      },
     ...
   }
 }
```

Configuring Incoming Channel

For performance improvement tuning **uncomment** or add the following in the **alarm-consumer-static-config.yaml** file to override the default configuration:

- Edit **max.poll.interval.ms** to increase or decrease the delay between invocations of **poll()** while using the consumer group management.
- Edit **max.poll.records** to increase or decrease the maximum number of records returned in a single call to **poll()**.

```
mp.messaging:
  incoming:
    toFaultChannel:
#      max.poll.interval.ms: 300000
#      max.poll.records: 25
    toRetryChannel:
#      max.poll.interval.ms: 300000
#      max.poll.records: 25
    toDltChannel:
#      max.poll.interval.ms: 300000
#      max.poll.records: 100
```

Impact Analysis API

The impact analysis API is as follows:

```
impactAnalysis:
  url: http://localhost:8084
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

Resource Type Mappings

The **TMF639ResourceType-mappings.yaml** file provides mapping from protocol specific to TMF639 sub-resource types supported by UIM. This mappings file should be updated only when the alarm event is not sent with resource type values supported by the UIM.

The **alarmedObject.@referredType** element value should be representing the TMF639 sub-resource type, which is supported by UIM. You must use this resource mapping extensibility only when the assurance system does not send the resource types which UIM can understand.

For example: Sending some protocol specific object type. The Corba protocol has object types as OT_EQUIPMENT, OT_MANAGED_ELEMENT, and so on.

When an alarm event does not contain UIM's TMF639 sub-resource types, map the protocol-native resource type to its TMF639 sub-resource type in the **TMF639ResourceType-mappings.yaml** file and upgrade the alarm consumer service.

The TMF639 resource types supported in alarm consumer are: PhysicalDevice, Equipment, PhysicalPort, and DeviceInterface.

The **TMF639ResourceType-mappings.yaml** is provided in **$SPEC_PATH/$PROJECT/$INSTANCE/config/ata/alarm-consumer** for extensibility. This file is available with some out-of-the-box default mappings as follows:

```
deviceTypeMapping:
  PhysicalDevice:
    - OT_MANAGED_ELEMENT
  Equipment:
    - OT_EQUIPMENT
    - CHASSIS
    - BACKPLANE
    - MODEL
    - RACK
    - SHELF
    - CARD
  PhyiscalPort:
    - OT_PHYSICAL_TERMINATION_POINT
    - PORT
    - PTP
  DeviceInterface:
    - OT_CONNECTION_TERMINATION_POINT
    - CTP
  Unknown:
```

Customizing Device Mapping

By default, the device or sub-device is found by nativeEmsName (using the value from the **alarmedObject.id**). Configuration can be updated to check the device or sub-device by other fields (such as **Id** or **deviceIdentifier**) when the alarm event is having id or **deviceIdentifier** values as part of the **alarmedObject.id** element.

In order to use different lookup fields, configure the **deviceMappings.inventory.lookupFields** sub-structure accordingly in the **alarm-consumer-static-config.yaml** file and upgrade the ATA service.

A sample file is provided in **$SPEC_PATH/$PROJECT /$INSTANCE/config/ata/alarm-consumer** location. If you are modifying it for the first time, rename the **alarm-consumer-**

Copyright © 2023, 2026, Oracle and/or its affiliates.

static-config.yaml.sample file to **alarm-consumer-static-config.yaml** and update the values accordingly. The supported lookup fields are **name**, **id**, and **nativeEmsName**. A sample sub-structure is as follows:

```
deviceMapping:
  inventory:
    lookupFields: # The lookup is done according to the provided order.
Supported values are name, id & nativeEmsName
    # - name
    # - id
    - nativeEmsName #default
  customizeDeviceLookup:
    enabled: false
```

The above YAML configuration is used to change the device mapping.

Table 11-1 Device Mapping Fields

Field	Description
deviceMapping.lookupFields	This is an array field that can have only the values : **name**, **id**, and **nativeEMSName**. **Note**: The valid values from 1.3.0 are **name**, **id**, **deviceidentifier**, and **nativeEMSName**. There are names of the fields in UIM Entity which will be used to search the device/sub-device what is mentioned in the **alarmedObject.id** field. The **nativeEmsName** field is default. That means, the first part of the '::' of **alarmedObject.id** field by default is searched with **nativeEmsName**. The order of the array is followed. Therefore, if array is updated in **id**, **nativeEmsName**, or **name** fields, the alarm consumer will take the first part of '::' of **alarmedObject.id** field and then search it in database with the corresponding id, since the array first element is id. Only the first three entries of the array are considered for searching. That means, for **name**, **id**, and **nativeEmsName** settings, it will search with the **nativeEmsName** first and if not found then **name**, if not found then **id**. Once a device is found, no further matching will be performed. In case no device is found or multiple devices found, see "Fallout Events Resolution for Alarm Consumer" for more information.
deviceMapping.customizeDeviceLookup.enabled	This is a Boolean value. The default value is **false**. If it is **true**, the alarm consumer enables extensibility to its user to provide Groovy script, which should return a single value by processing either **alarmedObject** or alarm (TMF-642). This single value which is expected to be returned from the Groovy is used to match in the database. The sample Groovy script is mentioned in the alarmed object extension sections.

Copyright © 2023, 2026, Oracle and/or its affiliates.

> **Note**
>
> In the previous release, **deviceMapping.lookupFields** was mentioned to have possible values like deviceIdentifier, ipv4, and ipv6 also. From 2.0.0.0, alarm on sub-node is supported. The **deviceMapping.lookupFields** does not support deviceIdentifier, ipv4, and ipv6. Valid values from 1.3.0 are **name**, **id**, **nativeEmsName** only.

```
import groovy.json.JsonSlurper
/**
 * The default delimiter is which is available via method argument named
"delimiter".
 * The "alarmedObject" parameter is extracted from alarmedObject sub-section
as String from the Alarm.
 * The "alarm" parameter is the complete alarm information received as String.
 * The return type must be type of Map of String as key and value.
 */
def getDeviceIds(delimeter,alarmedObject,alarm){
  def jsonSlurper = new JsonSlurper()
  def alo = jsonSlurper.parseText(alarmedObject)
  def aloId = alo.id
  def referredType = alo.'@referredType'
  def device = aloId.split(delimeter)
  def deviceInfo = [:]
  //Custom implementation starts.  The following is default implementation
which return the keys.
  //node and subNode should be the name of which will be searched in ATA/
Inventory databases.
  if (device.size() == 2) {
    //node-name
    deviceInfo["node"] = device[0]
    //subNode-name
    deviceInfo["subNode"] = device[1]
  } else {
     deviceInfo["node"] = device[0]
     deviceInfo["subNode"] = ""
  }
  //The referredType must match with TMF639ResourceType-mappings.yaml mapping
file.  Blank value considered as PhysicalDevice
  deviceInfo["referredType"] = referredType
  //Custom implementation ends
  return deviceInfo;
}
```

Alarmed Object Extension

If the **alarmedObject** sub-structure has different values or format than the above sub-sections, the provided Groovy file has to be modified to parse and return the identifier. This Groovy custom code runs when the **deviceMapping.customizedDeviceLoop.enabled** element value is configured to **true** in the **alarm-consumer-static-config.yaml** file.

Update the out-of-the-box provided Groovy code and the system returns the node/sub-node identifier and referredType values from this Groovy file. This Groovy file is provided in **$SPEC_PATH/$PROJECT/$INSTANCE/config/ata/alarm-consumer** location. Enable the

Copyright © 2023, 2026, Oracle and/or its affiliates.

deviceMapping.customizedDeviceLoop.enabled value to **true** and update the alarm consumer service.

The sample implementation is as follows:

```
/*
* Copyright (c) 2025. Oracle and/or its affiliates. All rights reserved.
*/
import groovy.json.JsonSlurper
/**
 * The default delimeter is which is available via method argument named
"delimeter".
 * The "alarmedObject" parameter is extracted from alarmedObject sub-section
as String from the Alarm.
 * The "alarm" parameter is the complete alarm information received as String.
 * The return type must be type of Map of String as key and value.
 */

def getDeviceIds(delimeter,alarmedObject,alarm){
  def jsonSlurper = new JsonSlurper()
  def alo = jsonSlurper.parseText(alarmedObject)
  def aloId = alo.id
  def referredType = alo.'@referredType'
  def device = aloId.split(delimeter)
  def deviceInfo = [:]
  def alarmObj = jsonSlurper.parseText(alarm)
  def alarmDetails = alarmObj.'alarmDetails'
  def alarmDetailsObj = jsonSlurper.parseText(alarmDetails)

    //Custom implementation starts.  The following is default implementation
which return the keys.
    //node and subNode should be the name of which will be searched in ATA/
Inventory databases.
    if( (referredType=="" ||
referredType.toString().equalsIgnoreCase("unknown") ) &&
alarmDetailsObj.containsKey("trap")) { //For the SNMP Scenarios
        if (device.size() == 2) {
            //node-name
            deviceInfo["node"] = device[0]
            //subNode-name
            deviceInfo["subNode"] = device[0]+"::"+device[1]
            deviceInfo["referredType"] = "Unknown"
        } else {
            deviceInfo["node"] = device[0]
            deviceInfo["subNode"] = ""
            deviceInfo["referredType"] = "PhysicalDevice"
        }
    }
    else { //For CORBA Scenarios
        //The referredType must match with TMF639ResourceType-mappings.yaml
mapping file.  Blank value considered as PhysicalDevice
        deviceInfo["referredType"] = referredType

        if (device.size() == 2) {
            //node-name
            deviceInfo["node"] = device[0]
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
            //subNode-name
            deviceInfo["subNode"] = device[1]
        } else {
            deviceInfo["node"] = device[0]
            deviceInfo["subNode"] = ""
        }
    }
    //Custom implementation ends
  return deviceInfo;
}
```

> **Note**
>
> The Groovy script (which was available on previous version) is not compatible with the current version. Consider this to be a new script which supports the alarm on sub-node. The previous Groovy script logic has to be written in to this new Groovy file.

Alarm Event Enrichment Extension

Use the alarm event enrichment extension to customize the input alarm event. For example: Populating a value for the missing element, transforming an element value, or any other change to the input alarm.

The sample in the following Groovy extension file (**EnrichAlarm.groovy**), sets the `reportingSystemId` element value to **97** characters if the size is greater than **100** characters.

The Groovy extension code is run when the `enrichAlarm.enabled` value is configured to **true** in the **alarm-consumer-static-config.yaml** file.

To customize further, update the out-of-the-box provided Groovy code and return the modified event JSON object as a string from the Groovy file. The Groovy file is available at **$SPEC_PATH/$PROJECT/$INSTANCE/config/ata/alarm-consumer**. Enable the `enrichAlarm.enabled` value to **true** and upgrade the alarm consumer service after customizing.

The default sample Groovy implementation is as follows:

EnrichAlarm.groovy

```
/*
* Copyright (c) 2025. Oracle and/or its affiliates. All rights reserved.
*/

/**
 * The enrich Groovy method has to be updated for any custom changes need to
the alarm event. The input to this method is JSON String object
representation of TFM642/688 alarm event.
 * The alarm consumer service, consumes this alarm event from the Kafka
Message Bus which is produced by the assurance system.  This method is
invoked before any processing in the alarm consumer.
 * This Groovy method has a sample on trimming the reportingSystemId value to
97 characters when the size if greater than 100 characters.
 */
import groovy.json.JsonBuilder
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
import groovy.json.JsonSlurper

def enrich(alarmEvent) {
    def jsonS = new JsonSlurper()

    def alarmEventObj = jsonS.parseText(alarmEvent)
    def alarmEventType = alarmEventObj.eventType //Null case already handled
before

    if(!alarmEventObj.containsKey("event") || !alarmEventObj["event"])
        return alarmEvent //If null, return nothing to change

    switch(alarmEventType) {
        case "AlarmCreateEvent":
        case "AlarmAttributeValueChangeEvent":
        case "AlarmDeleteEvent":
            if(!alarmEventObj["event"].containsKey("alarm") || !
alarmEventObj["event"]["alarm"]
                || !alarmEventObj["event"]
["alarm"].containsKey("reportingSystemId") || !alarmEventObj["event"]["alarm"]
["reportingSystemId"])
                return alarmEvent //If null, return nothing to change
            if(alarmEventObj["event"]["alarm"]
["reportingSystemId"].length()>100)//If length>100, trim it to 97 chars and
append "..." to indicate it's trimmed
                alarmEventObj["event"]["alarm"]["reportingSystemId"] =
alarmEventObj["event"]["alarm"]["reportingSystemId"].substring(0,97) + "..."
            break
        case "ClearAlarmCreateEvent":
            if(!alarmEventObj["event"].containsKey("clearAlarm") || !
alarmEventObj["event"]["clearAlarm"]
                || !alarmEventObj["event"]
["clearAlarm"].containsKey("clearedAlarm") || !alarmEventObj["event"]
["clearAlarm"]["clearedAlarm"])
                return alarmEvent //If null, return nothing to change
            for(individualClearedAlarm in alarmEventObj["event"]["clearAlarm"]
["clearedAlarm"]) {
                if(!individualClearedAlarm.containsKey("reportingSystemId")
|| !individualClearedAlarm["reportingSystemId"])
                    continue;

if(individualClearedAlarm["reportingSystemId"].length()>100)//If length>100,
trim it to 97 chars and append "..." to indicate it's trimmed
                    individualClearedAlarm["reportingSystemId"] =
individualClearedAlarm["reportingSystemId"].substring(0,97) + "..."
            }
            break
    }

    return new JsonBuilder(alarmEventObj).toPrettyString()
}
```

Mounting Groovy Scripts To Alarm Consumer

To mount Groovy scripts to alarm consumer pod:

Copyright © 2023, 2026, Oracle and/or its affiliates.

1. Edit the script in **$SPEC_PATH/$PROJECT/$INSTANCE/config/ata/alarm-consumer/** DeviceMapping.groovy location.
2. Upgrade ATA instance as follows:

```
$COMMON_CNTK/scripts/upgrade-applications.sh -p $PROJECT -i $INSTANCE -
s $SPEC_PATH -a ata
```

Service Impact Analysis Customer Mappings

See "Impact Analysis Customer Mappings" for more information.

Roles Required for Accessing Service Impact Analysis

For information on roles required for accessing Service Impact Analysis, see "About Authentication".

Deploying Service Impact Analysis Instance

To deploy a Service Impact Analysis instance in your environment using the scripts that are provided with the toolkit, run the following command to create an instance after updating the **applications-base.yaml**, **app-ata.yaml**, and configuring Service Impact Analysis:

```
$COMMON_CNTK/scripts/upgrade-applications.sh -p sr -i quick -s $SPEC_PATH -a
ata
```

Managing Service Impact Analysis Instance

The SIA instance consists of alarm-consumer and impact-analysis-api services. Update the corresponding sections in the **app-ata.yaml** file and follow the steps mentioned in the following sections of "Deploying the Active Topology Automation Service":

- Upgrading the ATA Instance
- Deleting and Recreating a ATA Instance
  To delete only Service Impact Analysis, update the respective **replicaCount** to **0** and upgrade the instance.
- Restarting the ATA Instance

Managing Service Impact Analysis Logs

To customize and enable logging, update the logging configuration files for the application as follows:

1. Customize **impact-analysis-api** service logs:
   - For service level logs, update the **$SPEC_PATH/$PROJECT/$INSTANCE/config/ata/impact-analysis-api/logging-config.xml** file.
   - For Helidon-specific logs, update the **$SPEC_PATH/$PROJECT/$INSTANCE/config/ata/impact-analysis-api/logging.properties** file. By default, the console

Copyright © 2023, 2026, Oracle and/or its affiliates.

handler is used. You can provide filehandler, uncomment the following lines, and provide the project and instance names for location to save logs.

```
handlers=io.helidon.common.HelidonConsoleHandler,java.util.logging.FileHand
ler
java.util.logging.FileHandler.formatter=java.util.logging.SimpleFormatter
java.util.logging.FileHandler.pattern=/logMount/sr-quick/ata/ata-api/logs/
ImpactAnalysisJULMS-%g-%u.log
```

2. Customize **alarm-consumer** service logs as follows:
   - For service level logs, update the **$SPEC_PATH/$PROJECT/$INSTANCE/config/ata/alarm-consumer/logging-config.xml** file.
   - For Helidon server logs, update the **$SPEC_PATH/$PROJECT/$INSTANCE/config/ata/alarm-consumer/logging.properties** file.
3. Once the log configuration files are updated, upgrade the ATA instance. The sample upgrade script is as follows:

```
$COMMON_CNTK/scripts/upgrade-applications.sh -p sr -i quick -s $SPEC_PATH -
a ata
```

Alternate Configuration Options

See "Deploying the Active Topology Automation Service" for more information.

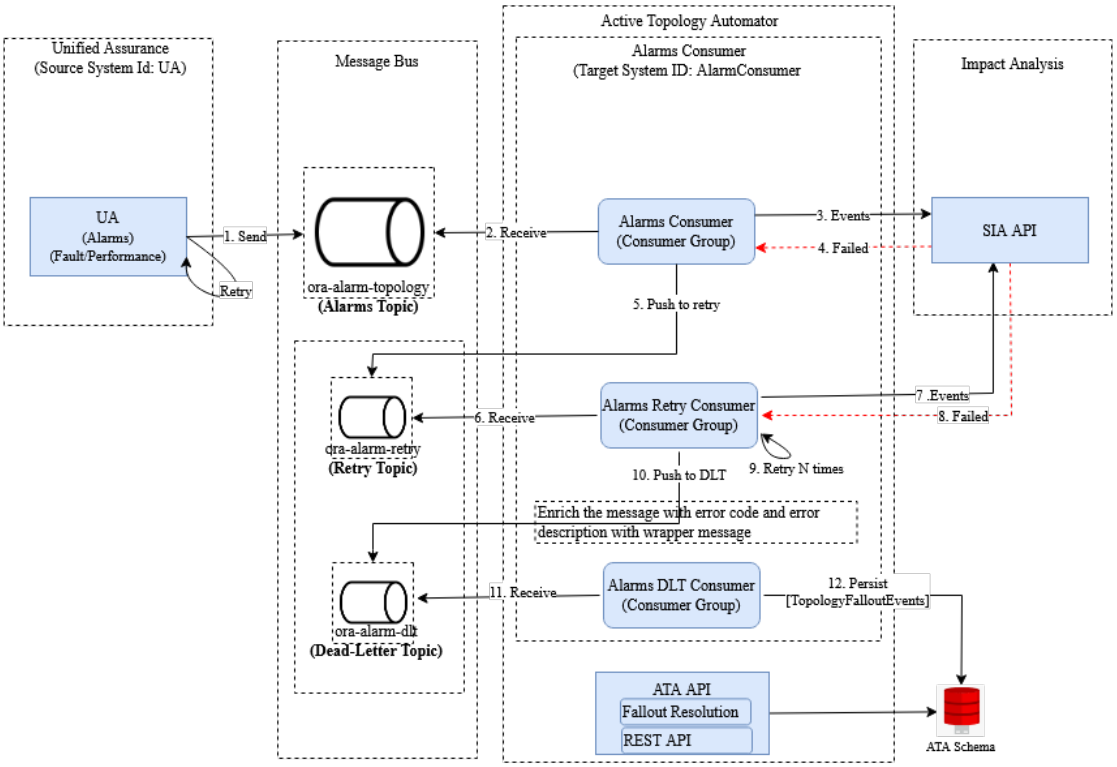
See "Fallout Events Resolution for Alarm Consumer" to resolve the fallout events for the alarms.

Fallout Events Resolution for Alarm Consumer

The following image illustrates an alarm event (or message) processing flow in alarm consumer.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Figure 11-1 Process Flow of Fallout Events Resolution for Alarm Consumer



Troubleshooting the Alarm Fallouts

Alarm fallouts are the alarms that could not be processed in alarm consumer because of the exceptions or errors occurred during processing of the alarm. Following are the major fallout scenarios identified:

- Incoming alarm has an invalid JSON structure or invalid TMF642 structure.
- No device is found to which the incoming alarm can be mapped.
- Multiple devices are found for the incoming alarm.
- Processed alarm could not be forwarded to SIA API.

In the mentioned fallout scenarios, alarm-consumer is configured to retry the processing of the same alarm. This helps to address the possible intermittent issues such as connectivity and temporary data unavailability. In case the retry processing of the alarm is a fallout scenario, the details of the fallout information along with the alarm information will be stored into database. The persisted fallout alarms can be reviewed manually. In the process of reviewing of the fallout alarm, the reviewer can add modification or correction to the alarm data and send the fallout alarm for reprocessing.

> **Note**
>
> All fallout alarms are persisted. The alarms that have invalid JSON structure or invalid TMF642 structure will not be persisted. These non-persisted fallout alarms are dropped from the alarm-consumer and alarm details cannot be verified later.

Copyright © 2023, 2026, Oracle and/or its affiliates.

The **TARGETSYSTEMID** for all alarm fallouts for the alarm consumer will be **AlarmConsumer**. Therefore, while running a fallout resolution, the target service value will be used as **AlarmConsumer**.

The new Message Reconciliation application helps you in resolving the fallout events. See *Active Topology Automator and Service Impact Analysis User's Guide* for details on the Message Reconciliation UX flow.

> **Note**
>
> Oracle recommends to use this Message Reconciliation user interface for resolving your fallout events.

The alarm consumer fallout alarms can be used for reprocessing, using the Message Reconciliation UX or directly using the fallout resolution end-points provided as part of ATA-API. Oracle recommends you to use the fallout resolution end-points directly for any headless automation:

- To list all alarm consumer fallout alarms:
  - **Method** - GET
  - **URI** - `/topology/v2/fallout/events?targetService=AlarmConsumer`
- To find a specific fallout alarm:
  - Method - GET, **URI** - /topology/v2/fallout/events/eid/<eid value>
- Update fallout alarm data:
  - **Method** - PUT
  - **URI** - `/topology/v2/fallout/events/eid/<eid value>`
  - **Request Body** - Includes the fallout event details found using the above mentioned APIs.
- To send for reprocessing:
  - **Method** - POST
  - **URI** - `/topology/v2/fallout/events/resubmit?state=PENDING&action=NEW&targetService=AlarmConsumer`

Copyright © 2023, 2026, Oracle and/or its affiliates.

12

Dynamic Attribute Mapping between UIM and ATA

This chapter describes how to perform the Dynamic Attribute mapping between UIM and ATA.

Dynamic Data Mapping from UIM

The dynamic data mapping takes advantage of UIM characteristics and provides maximum flexibility for mapping fields from UIM to the topology model.

The dynamic data mapping:

- Does not require any additions, updates, migrations, or deployments of your existing specifications.
- Guarantees the value is set correctly and does not require a user to select the correct value.
- Allows ATA to support data extensions to the topology model without an upgrade.
- Vertex and Edge Labels or Properties in ATA may require different names than Characteristics, or Attributes or Roles in the implemented UIM model.
- These items are supported through dynamic data mapping.

The examples are:

- UIM has a 'Vendor' attribute on the Logical Device and Equipment Specifications but some users have added 'manufacturer' to their Physical Device Specifications.
- Some vertices are not identified specifically in UIM such as Domain and Service Type. These values are implied based on the '5G' cartridge or the 'FTTx' cartridge but are not specifically identified on the entity.

Prerequisites for Dynamic Data Mapping from UIM

The prerequisites are as follows:

- The following configuration files are required:
  - topologyAttributeMapping.json
  - topologyRoleMapping.json
  - topologySpecificationMapping.json.
- For traditional UIM, these files exist in the <domain>/UIM/config/topologyMappings directory. For a cloud native environment, the files exist in the $SPEC_PATH/$PROJECT/$INSTANCE/config/uim/topologyMappings directory after assembling specifications.
- Files with these names plus the extension **.sample** are provided.
- Prior to migration, the correct configurations must be provided. Else, the data will not be mapped correctly to ATA.
- If the file does not exist an error occurs during UIM entity creation.

Copyright © 2023, 2026, Oracle and/or its affiliates.

- If you want to skip this process, you can remove the **.sample** extension and proceed with the default settings.

Planning the Mapping

You can avoid mapping all characteristics to ATA if you have a large set of characteristics. Determine which characteristics are helpful for developing the proper answers to your topology questions.

For example, if you want to query all **Router** specifications:

1. Identify all characteristics and specification names that identify a **Router**.
2. Map those values to the **PG_DEVICE.NODECATEGGORY** column.
3. Run a query `'Node Category' = 'Router'` to display the topology for all your routers.

Another example is **Vendor**. If you identify your <*vendor1*> devices by specification name or by characteristics, map them to the **PG_VENDOR Vertex**. Once this is setup, you can query all your <*vendor1*> Devices or combine it with the previous examples and query all your vendor Routers. See *ATA Model Documentation* for the full set of columns and vertices that are available.

Characteristics that are mapped as **Properties** are displayed when you select **Properties** on the **Resource Drawer** from the ATA canvas. If you have a large set of characteristics that are mapped to **Properties**, an alternate flow is recommended. If you use **More Info** from **Resource Drawer** and visualize the characteristics from the Resource Summary page, you see all attributes without impacting the performance.

Ruleset-Based Solution for Generating Dynamic JSON Mapping Files

The set of JSON files used for UIM to ATA Data Migration and UIM-ATA integration are different.

To generate the Dynamic JSON Mapping files for UIM to ATA Data Migration and UIM-ATA integration:

1. The Ruleset expects the Excel file in the format specified in the sample Excel file. The sample Excel file is located at **UIM_SDK/cartridges/base/ora_uim_ata_DAMJson_cartproj.zip/doc/UIM-UTIA_Mig_Json_v2.xls**.
2. Change the content of the Excel file as required.
3. Deploy this ruleset in your UIM environment: **UIM_SDK/cartridges/base/ora_uim_ata_DAMJson_cartproj.zip/cartridgeBin/ora_uim_ata_DAMJson.jar**.
4. Add the following to `importExport.properties` in **<DOMAIN_HOME>/UIM/config/importExport.properties**:

   ```
   import.fileUploadWhiteListMimeTypes=text/plain,text/csv,application/
   zip,application/x-zip-compressed,application/vnd.openxmlformats-
   officedocument.spreadsheetml.sheet,application/vnd.ms-excel
   ```

5. After successful deployment of the cartridge, navigate to **UIM UI**, **Administration**, and then to **Execute Rule**.
6. From the **Ruleset** list, select **ATA_DAMGenerator**.
7. Click **Choose File** and select the Excel file that is in **.xls** format.

Copyright © 2023, 2026, Oracle and/or its affiliates.

8. Click **Process** at the top-right corner of the page.
The JSON Mappings files in the **DOMAIN_HOME>/UIM/config/topologymappings/migration_Jsons** path are created. There are two folders **migrationMappings** and **TopologyMappings** that contain the JSON files required for the UIM to ATA Data Migration and UIM-ATA integration respectively.

> **Note**
>
> - Files located in **migrationMappings** should be used during Dynamic Attribute Migration. See "Planning the Topology Migration" for more information.
> - Files located in **TopologyMappings** should be used during UIM-ATA integration. See "Dynamic Data Mapping from UIM" for more information.

9.

Mapping the Dynamic Data from UIM

To map the dynamic data from UIM, the following definitions are required:

- vertex: A node in the Topology Model, examples are Vendor, Domain, Technology, Network Type, Device, Location
- property: A column on every vertex and edge in the Topology model.
  - It supports JSON allowing for unlimited additional attributes.
  - Property is the name of the key used to store the value retrieved from the UIM attribute.
- properties: Is an array defining how individual attributes of an entity are to be stored in Topology schema.
- columnName: An existing column on a physical table in the Topology Model used to store the attribute.
- name: Maps to different entity classes and entity specification classes. For example: “LogicalDeviceDAO”, “EquipmentSpecificationDAO”, “PlaceSpecificationDAO”, “PropertyLocationDAO” and so on.

The following POST operation creates a logical device, you can see the relationships and properties with which the dynamic properties are supported.

POST: http://localhost:8080/vertex

Body:

```
{"entityId":<entityID>,"entityVersion":<entityVersion>,"businessObjectClass":"
LogicalDeviceDAO","id":"<ID>","name":"<name>","specName":"<specificationName>"
,"latitude":0.0,"longitude":0.0,"inventoryStatus":"INSTALLED","referenceId":<r
eferenceID>,"relationships":{"vendor":"<vendor>"},"properties":
{"deviceIdentifier":"<deviceIdentifier>"}}
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

> **Note**
>
> - In this example, the TopologyAttributesMapping.json file provides the instructions to ATA and the file is available in the UIM/config/topologyMappings directory.
> - The topologyAttributesMapping file is used to address hard coded attributes from UIM tables.
> - See topologyAttributesMapping.json for more information.

The POST operation tells the topology:

- Map LogicalDevice.deviceIdentifier to the property deviceIdentifer.
- Map LogicalDeviceSpecification.vendorName to the vertex = vendor
- This is based on the UIM ClassName, it works with any Class or specification that is topology-enabled.

You can add a role to the Logical Device from the list of roles that are configured in the TopologyRoleMapping.json file.

You can see that GET that the Logical Device tracks the **deviceIdentifier** in the **properties** column using:

GET: http://localhost:8080/vertex/typeid/1/referenceid/<refID>

```
{"businessObjectClass":"LogicalDeviceDAO","entityId":<entityID>,"entityVersion
":<entityVersion>,"id":"<versionID>","inventoryStatus":"INSTALLED","latitude":
0.0,"longitude":0.0,"name":"<name>","properties":
{"deviceIdentifier":"<deviceID>"},"referenceId":<referenceID>,"specName":"<spe
cificationName>"}
```

PUT: http://localhost:8080/vertex

```
{"businessObjectClass":"LogicalDeviceDAO","entityId":<entityID>,"entityVersion
":3,"id":"<ID>","inventoryStatus":"INSTALLED","latitude":0.0,"longitude":0.0,"
name":"<name>","properties":
{"deviceIdentifier":"<ID>","transmission":"Optical_Transmission"},"referenceId
":<refernceID>,"specName":"<specificationName>"}
```

In the body:

- The role “Optical_Transmission” is mapped to the property field with name = “transmission”.
- The role was given a name = “transmission” which was provided by the UIM admin.
- Add, update and delete are supported. This works for Equipment and Physical Device (any topology-enabled entity that supports roles).
- Roles can be mapped to properties, vertices or columns.

The rules to perform this are:

- The Vertex must exist: The mapping can be performed to multiple vertices and can have multiple values.

Copyright © 2023, 2026, Oracle and/or its affiliates.

- Property: There can be multiple properties. The UIM integrator is responsible for not having similar or misspelled values.
- ColumnName: A column can only have 1 value. The user is currently responsible for assuring this value is unique. It can be overlaid. This should be used for a queried attributes where an index is needed.
- The possible values of "columnName" are the following:
  - PG_DEVICE - [NODECATEGORY, MACADDRESS, IPV4, IPV4SUBNET, IPV6, IPV6SUBNET, ZONEID, DEVICEIDENTIFIER, NETWORKSTATUS, NODETYPE]
  - PG_LOCATION - [DISTRICT, PROVINCE, OPERATOR, CITY, STATE, POSTALCODE, COUNTRY, AREA, CIRCLE]
  - PG_COMMICATION - [FROMNODEDATA, TONODEDATA, RATECODE, TECHNOLOGY]
  - PG_NETWORK - [CATEGORY, SUBCATEGORY, TOPOLOGYTYPE, SUBTYPE]

> **Note**
>
> UIM currently supports city, state, country and postalcode attributes from the PropertyLocationDAO and PropertyAddressDAO. The street address or subunit (aptartment number, room number) are not supported.

The supported UIM classes are:

LogicalDeviceDAO, GeographicPlaceDAO, PhysicalDeviceDAO, NetworkDAO, NetworkEdgeDAO, EquipmentDAO, GeographicSiteDAO, PropertyLocationDAO

> **Note**
>
> This includes the corresponding supported specification classes.

The last configuration is **TopologySpecificationMapping.json**.

- The **related vertices** field automatically adds a relationship edge between any instance of the specification to the vertex with the provided name and value.
- A characteristic does not need to be added and set on the specification to be tracked in topology.
- This allows our current RI cartridges to be used without any modifications.
- The **characteristics** column works the same as roles.
- It automatically adds a relationship to a vertex, sets properties or sets a column value.
- Any current characteristics can be used. No changes are needed.

PUT: http://localhost:8080/vertex

```
{"entityId":<entityID>,"entityVersion":<entityVersion>,"businessObjectTypeId":
1,"businessObjectClass":"LogicalDeviceDAO","id":"<ID>","name":"<name>","specNa
me":"router","latitude":<latitude>,"longitude":<longitude>,"inventoryStatus":"
INSTALLED","isTopLevelNode":true,"nodeAvailable":true,"placeNode":false,"refer
enceId":<referenceID>,"createdUser":"test","lastModifiedUser":"test","relation
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
ships":{"vendor":"<vendor>","domain":"Ethernet"},"properties":
{"deviceIdentifier":"<deviceID>"}}
```

Customizing ATA Service Topology Configurations from UIM

From UIM, you can customize the ATA service topology configurations. You update the required properties, that customize the service topology configurations in ATA, from a JSON file: `ServiceTopologyConfiguration_<serviceSpecName>.json`. Each service has its own JSON configuration file.

For traditional UIM, this service topology configuration JSON file is located in *UIM_Home*/**config/topologyMappings**. For a cloud native environment, the JSON file exists in **$SPEC_PATH/$PROJECT/$INSTANCE/config/uim/topologyMappings**.

Service topology has an out of the box functionality to build a visualization for each service based on the resources available in the service configuration. You can customize the visualization to complete the end-to-end design as per your requirement.

For example:

- To add a custom node **Internet** to the end of the Broadband service and a custom edge to connect it to a router that is a resource on the service.
- To add a custom edge and label it **Inside Wiring** to show the connection from the CPE to the ONT.

The table lists the service topology configuration properties.

> **Note**
>
> - In the table below, nodes and edges that are added to support the topology visualization are referred to as **TOPOLOGYONLY** nodes or edges.
> - You can use the custom icons, customlabels, custom line styles, custom colors and exclude nodes which are redundant or not important.

Table 12-1 Properties in the ServiceTopologyConfiguration_<serviceSpecName>.json File

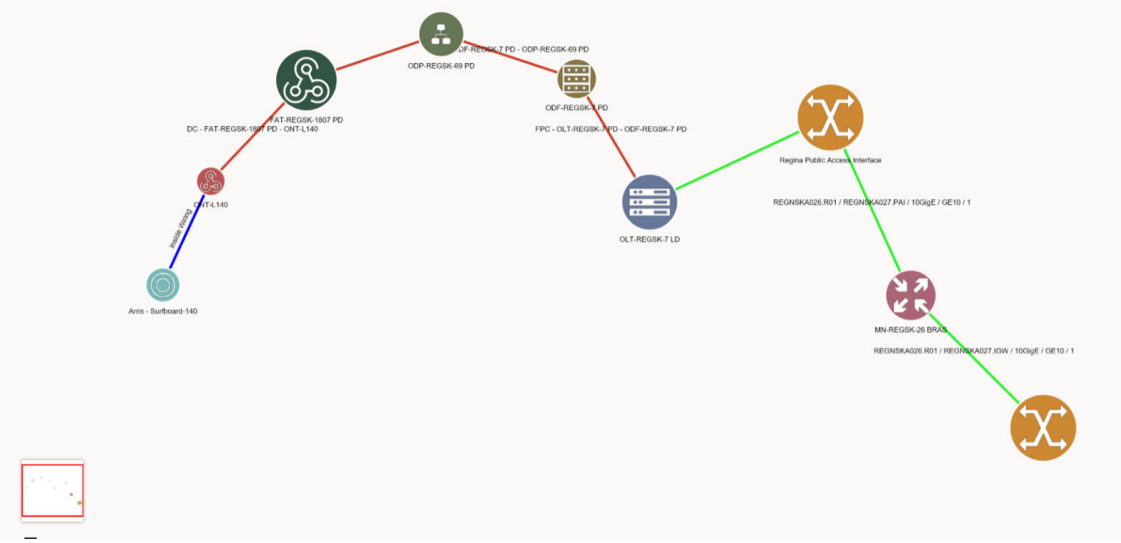
Property	Description
configurationItemLabel	This property sets the service item name for which a resource is assigned or referenced. Setting this property creates a node or edge for the resource. For any service item, this property is mandatory. For a **TopologyOnly** node, the item lable is `TopologyOnly + <uniqueId>`. The Id must be unique for all the entries in the service topology configuration JSON file.
configurationItemParent	This property sets the child service item. If the service item is not under a child service, you do not set this property.
customLabelClass	This property sets the custom class. Using this, you can customize the label of a node or an edge. This custom class overrides the values set by **getNodeLabel** and **getEdgeLabel** methods.
exclude	By default, all the service nodes and edges are added to the topology in ATA. To exclude any of these nodes and edges from being displayed, set this property to **true**.
icon	This property enables you to customize an icon.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Table 12-1 (Cont.) Properties in the ServiceTopologyConfiguration_<serviceSpecName>.json File

Property	Description
customType	This property sets the custom type for a node. For **TopologyOnly** nodes, set the **customType** value as `CustomDeviceDAO/CustomNetworkDAO/CustomLocationDAO/CustomNECDAO`.
sourceNode	This property sets the source node for a **TopologyOnly** edge. The value for this property can be the name or label of the node.
targetNode	This property sets the target node for a **TopologyOnly** edge. The value for this property can be the name or label of the node.
lineStyle	This property sets the line style. The available values are **solid**, **dot**, and **dash**.
lineColor	This property sets the line color.

The following figure shows the end-to-end view of a Broadband service using JSON customizations.




```
"FibreBroadband" : {
    "nodes": [
        {
            "configurationItemLabel" : "OLTPort", "exclude": "true"
        },
        {
            "configurationItemLabel" : "TransportNetwork", "exclude": "true"
        },
         {
            "configurationItemLabel" : "VoiceNetwork" , "exclude": "true"
        }
    ],
    "edges": [
        {
            "configurationltemLabel": "TOPOLOGYONLY31",
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
            "customLabelClass" : "oracle.communications.inventory.api-
custom.CustomLabelClass",
            "sourceNode" : "CPE",
            "targetNode": "ONT”,
                        "lineStyle": "SOLID",
                        "lineColor": "BLUE"
        }
    ]
}
```

Custom Labels

Custom labels require a custom class be deployed with your cartridge using Design Studio. An example of the **Custom Label** class used above.

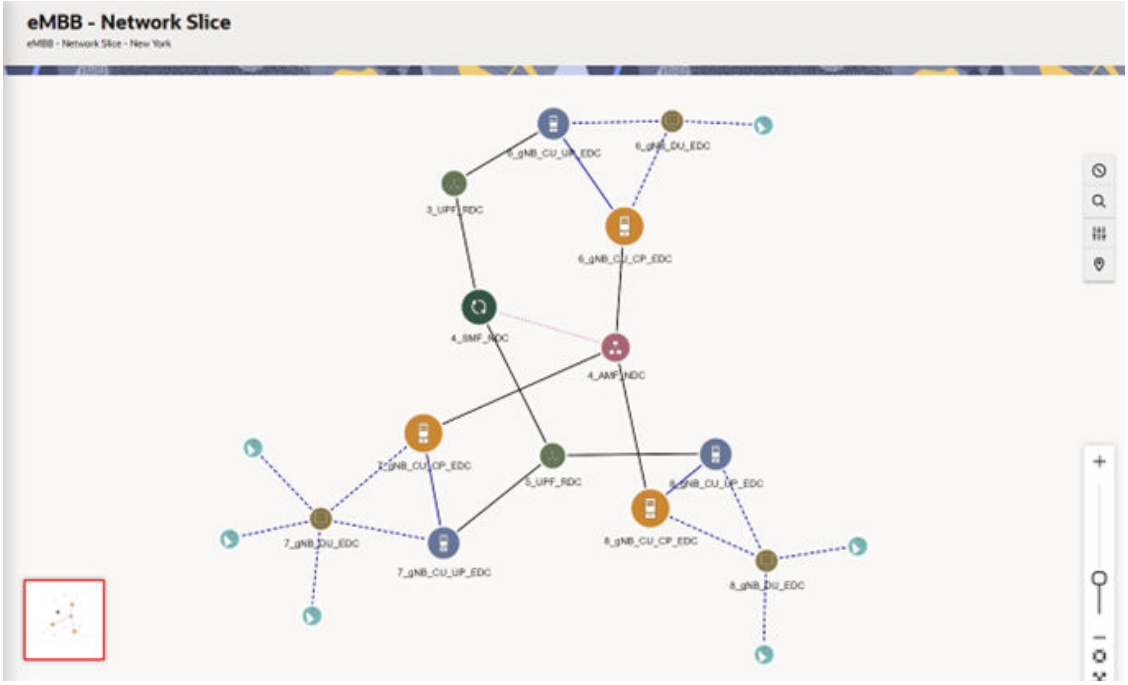
CustomLabelClass.java

```
package oracle.communications.inventory.api.custom;

import oracle.communications.inventory.api.entity.TopologyProfileEdge;

public class CustomLabelClass {

    public String getNodeLabel(TopologyProfileNode tpn) {
        System.out.println("tpn##########"+tpn);
        return "customNodeLabel";
    }
    public String getEdgeLabel(TopologyProfileEdge tpe) {
        System.out.println("tpe##########"+tpe);
        return "Inside Wiring";
    }

}
```

See the **ServiceTopologyConfiguration_NetworkSlice.json.sample** in **UIM_Home/config/topologyMappings** for the following Network Slice Service and Service Topology visualization.

Configuration	Resource Specification	Assignment/Reference Status	Resource
Core_Slice_Subnet	CORE NSS RFS	Referenced	3 - CORE_NetworkSliceSubnet-NY_A1_3
Core_Slice_Subnet	CORE NSS RFS	Referenced	4 - CORE_NetworkSliceSubnet-NY_A1_4
Service:CORE_NSS-NY_A1 - 1 - 1 - Se_4_1			
Characteristics			
Links			
AMF	AMF	Assigned	36 - 4_AMF_NDC
Characteristics			
NG11_Link	NG11_Link	Referenced	NYK.AMF_NDC2 / NYK.SMF_NDC3 / NG11 / 1
NgC_Link	NgC_Link	Referenced	NYK.AMF_NDC2 / STTN01NY.CP_EDC3 / NgC / 1
NgC_Link	NgC_Link	Referenced	NWRC01NY.CP_EDC3 / NYK.AMF_NDC2 / NgC / 1
NgC_Link	NgC_Link	Referenced	NWYR01NY.CP_EDC3 / NYK.AMF_NDC2 / NgC / 1
SMF	SMF	Assigned	37 - 4_SMF_NDC
Characteristics			
NG11_Link	NG11_Link	Referenced	NYK.AMF_NDC2 / NYK.SMF_NDC3 / NG11 / 1
NG4_Link	NG4_Link	Referenced	FLSH01NY.UPF_RDC3 / NYK.SMF_NDC3 / NG4 / 1
NG4_Link	NG4_Link	Referenced	BRKL02NY.UPF_RDC3 / NYK.SMF_NDC3 / NG4 / 1
ActivationDCTarget	Data Center	Referenced	75009 - NYNATIONAL_DC_008
SliceProfile	Profile	Referenced	34
SliceQoS	FiveQI Parameter Set	Referenced	1 - 5QI-1
SliceQoS	FiveQI Parameter Set	Referenced	2 - 5QI-2
vnfInfo			
vnfInfo			

Copyright © 2023, 2026, Oracle and/or its affiliates.



Impact Analysis Customer Mappings

By default, parties associated with the impacted entities are not considered as the impacted customers. Based on your requirement, you can track the impacted customers using the configuration file. By specifying these mappings, you can control which customers are included in the impact analysis results when specific entities are impacted by network events or changes.

To achieve this filtering, provide the configuration in a JSON file. The location of this file is as follows:

- **File location in UIM cloud native**: $SPEC_PATH/$PROJECT/$INSTANCE/config/uim/siaMappings/partyRoleMappings.json

  File location in a standalone installation: <DOMAIN_HOME>/UIM/config/siaMappings/partyRoleMappings.json

After making any changes to **partyRoleMappings.json**, restart UIM to make the mappings effective.

About JSON File Structure

The **partyRoleMappings.json** file is organized by entity type, with each entity type containing multiple entity specifications. For each entity specification, you can define the required party specifications and roles as impacted customers.

High-level Structure

```
{
  "EntityTypeDAO": [
    {
      "entitySpecification": "SpecificationName",
      "parties": [
        {
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
          "partySpecification": "PartySpecName",
          "roles": ["Role1", "Role2", ...]
        }
      ]
    }
  ]
}
```

Where:

- `EntityTypeDAO` represents the inventory entity type (for example, ServiceDAO, LogicalDeviceDAO).
- `entitySpecification` defines the specific type of entity within the category.
- `parties` contains an array of party specifications and their associated roles.
- `partySpecification` defines the type of party (for example, Individual, Business)
- `roles` lists the roles that the party must be considered as impacted.

Supported Entity Types

The following entity types are supported in the mapping file:

- **ServiceDAO**: Service entities (such as FibreBroadband, AccessService)
- **LogicalDeviceDAO**: Logical device entities (such as routers, switches)
- **PhysicalDeviceDAO**: Physical device entities (such as hardware devices)
- **EquipmentDAO**: Equipment entities (such as chassis, cards)
- **PipeDAO**: Connectivity entities (such as links, channels)

> **Note**
>
> Adding, viewing, editing, or deleting party associations for Logical Device, Physical Device, Equipment, Pipe, and Connectivity entities is not currently supported in UIM. However, party associations for these entities can be performed using Bulk Loader or APIs.

Examples for Impact Analysis Customer Mappings

Example: Service Entity Mappings

```
"ServiceDAO": [
  {
    "entitySpecification": "FibreBroadband",
    "parties": [
      {
        "partySpecification": "Individual",
        "roles": ["Customer"]
      }
    ]
  }
]
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

When a **FibreBroadband** service is impacted, the system identifies parties with specification `Individual` that have the role `Customer` associated with this service. These parties are then included as impacted customers in the analysis results.

Example: Multiple Roles for a Party Specification

```
"LogicalDeviceDAO": [
  {
    "entitySpecification": "Ciena 6500",
    "parties": [
      {
        "partySpecification": "Individual",
        "roles": ["Customer", "ServiceOwner", "TechnicalContact"]
      }
    ]
  }
]
```

For `Ciena 6500` logical device, the individuals with any of the roles `Customer`, `ServiceOwner`, or `TechnicalContact` will be considered as impacted parties.

Example: Multiple Party Specifications

```
"EquipmentDAO": [
  {
    "entitySpecification": "Ciena 6500-2 Chassis",
    "parties": [
      {
        "partySpecification": "Individual",
        "roles": ["Customer"]
      },
      {
        "partySpecification": "Business",
        "roles": ["EnterpriseCustomer", "ManagedServiceClient"]
      }
    ]
  }
]
```

For `Ciena 6500-2 Chassis` equipment the following roles will be considered as impacted parties.:

- The `Individual` parties with `Customer` roles.
- The `Business` parties with `EnterpriseCustomer` or `ManagedServiceClient` roles.

Configuring the Mapping File

Adding a New Entity Specification

To add a new entity specification to the mapping file:

1. Identify the required entity type (ServiceDAO, LogicalDeviceDAO, and so on).
2. Add a new entry with the entity specification name.
3. Define the party specifications and roles that should be associated with it.

Copyright © 2023, 2026, Oracle and/or its affiliates.

An example for adding a new GPON service mapping is as follows:

```
"ServiceDAO": [
  {
    "entitySpecification": "GPONService",
    "parties": [
      {
        "partySpecification": "Individual",
        "roles": ["Customer", "Subscriber"]
      },
      {
        "partySpecification": "Business",
        "roles": ["EnterpriseCustomer"]
      }
    ]
  }
]
```

Modifying Existing Mappings

To modify an existing mapping:

1. Locate the entity specification in the file.
2. Add or remove party specifications and roles as needed.
3. After making any changes to **partyRoleMappings.json**, restart UIM to get the changes updated.

An example for updating an existing mapping to add a new role is as follows:

```
// Original mapping
"PipeDAO": [
  {
    "entitySpecification": "DWDM_Optical_Fiber",
    "parties": [
      {
        "partySpecification": "Individual",
        "roles": ["Customer"]
      }
    ]
  }
]

// Updated mapping with new role
"PipeDAO": [
  {
    "entitySpecification": "DWDM_Optical_Fiber",
    "parties": [
      {
        "partySpecification": "Individual",
        "roles": ["Customer", "ServiceUser"]
      }
    ]
  }
]
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

13

Data Migration between UIM and ATA

This chapter describes how to perform the data migration between UIM and ATA.

Planning the Topology Migration

In preparation for implementing ATA, you must set up the topology migration and UIM to topology configuration. The UIM to topology migration extracts and loads necessary information from UIM into the topology graph model consisting of vertices and edges. Following the Database per Service Micro service Design Pattern, the topology graph resides in a Pluggable Database (PDB) container separated from the UIM database.

The migration consists of the following:

- Index Rebuilding: The index rebuilding consists of re-creating indexes on tables with migrated data, dropping the temporary tables created during migration and renaming the tables with migrated data to actual topology tables.

Data Migration Approaches

You can follow the following approaches for data migration:

- Data Migration through Database Link: Database Link (DBLink) is created from ATA schema to UIM schema.
- Data Migration through Read Access on UIM schema: ATA schema is set up within the same PDB as that of the UIM schema. ATA schema user is granted with **SELECT** (read access) along with the tables owned by UIM schema user. Data dump files are created for the migrated topology data. These dump files are then imported in the target PDB where the ATA schema will be placed.

The prerequisites are:

- Add DATAFILE to increase the TABLESPACE available (SYSTEM by default) for the ATA schema user. Preferably one-fourth the size of UIM schema.
- Data Migration to custom tablespace can be achieved by making the custom tablespace as the default tablespace for the ATA schema user.
- Cancel or complete all **In Progress** BIs that are associated with devices, equipment, connectivities, and pipes.

The Migration Steps are as follows:

1. Build Characteristics tables for the following topology enabled entities such as Equipment, Logical Device, Network, Network Edge, Physical Device, Pipe and Place. These **<ENTITY>_CHAR_MIG** tables are used to store all characteristics on each entity which are used during Dynamic Attribute Migration and Customizing Topology JSON files. Build **<ENTITY>_CHAR_MIG** tables:
    - Open a command line window and login to SQL*Plus for the UIM database.
    - Run the following SQL scripts providing the full path of the files. For example, use the @scriptFileName command where scriptFileName is the full path and name of the file.

Copyright © 2023, 2026, Oracle and/or its affiliates.

  - $WORKSPACEDIR/ata-builder/migration_scripts/Char_Mig_tables/CREATE_CHAR_MIG_TABLE.sql
  - $WORKSPACEDIR/ata-builder/migration_scripts/Char_Mig_tables/MIGRATION_CHAR1.sql
  - $WORKSPACEDIR/ata-builder/migration_scripts/Char_Mig_tables/MIGRATION_CHAR2.sql
  - $WORKSPACEDIR/ata-builder/migration_scripts/Char_Mig_tables/MIGRATION_CHAR3.sql

- To verify if the scripts ran successfully, you can verify that the UIM schema includes the following tables:
  - EQUIPMENT_CHAR_MIG
  - LOGICALDEVICE_CHAR_MIG
  - NETWORK_CHAR_MIG
  - NETWORKEDGE_CHAR_MIG
  - PHYSICALDEVICE_CHAR_MIG
  - PIPE_CHAR_MIG
  - PLACE_CHAR_MIG
  - CHARACTERISTICS_TABLE_MAPPING_MIG

> **Note**
>
> You can perform this step for any of the data migration approaches.

2. The Topology schema user account must have the following privileges:
   - CREATE JOB
   - CREATE SESSION
   - ALTER SYSTEM
   - CREATE DATABASE LINK
   - CREATE PROCEDURE
   - CREATE SEQUENCE
   - CREATE TABLE
   - CREATE TYPE
   - UNLIMITED TABLESPACE
   - CREATE JOB

   These above privileges are sufficient for Approach 1, however for Approach 2:
   - Create **SYNONYM**.
   - Grant **SELECT** permission to all the tables owned by UIM schema user and ATA schema user.

```
CREATE PROCEDURE grant_select(
    username VARCHAR2,
    grantee VARCHAR2)
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
AS
BEGIN
    FOR r IN (
        SELECT owner, table_name
        FROM all_tables
        WHERE owner = username
    )
    LOOP
        EXECUTE IMMEDIATE
            'GRANT SELECT ON '||r.owner||'.'||r.table_name||' to ' ||
grantee;
    END LOOP;
END;
    “username” - UIM Schema User
    “grantee” - ATA Schema User within the same PDB.
```

3. Static Attribute Migration:
   - Open a command line window and login to SQL*Plus for the Topology database.
   - **Approach 1**:
     - Migrate the static attributes data by running $WORKSPACEDIR/ata-builder/migration_scripts/data_migration_script_using_dblink.sql
     - The following input arguments are expected:
       * UIM schema username
       * UIM schema password
       * Database Hostname
       * Database port number
       * Database Service name
       * Commit Size(Optional – 50000(Default))
       * Maximum number of parallel processes(Optional – 5(Default))
       * Wait Time(Optional – 2(Default in seconds))
   - **Approach 2**:
     - Migrate the static attributes data by running $WORKSPACEDIR/ata-builder/migration_scripts/data_migration_script_using_localCopy.sql
     - The expects the following input arguments:
       * UIM schema username with in the PDB
       * Commit Size(Optional – 50000(Default))
       * Maximum number of parallel processes(Optional – 5(Default))
       * Wait Time(Optional – 2(Default in seconds))

   Where:
   - Commit Size: The number of records handled by a single process.
   - Maximum number of parallel processes: Depends on number of CPUs available.
   - Wait Time: Waiting interval after which the listener checks for the availability of jobs.

Copyright © 2023, 2026, Oracle and/or its affiliates.

> **Note**
>
> Wait until all background jobs for migrating capacity information are complete before proceeding to the next step. You can verify this by running the following command in the ATA schema:
>
> ```
> SELECT * FROM USER_SCHEDULER_JOBS WHERE STATE = 'RUNNING' and
> JOB_NAME LIKE 'POPULATE_PG_CAPACITY_%';
> ```

4. Enter the following UIM Schema details in **$WORKSPACEDIR/ata-builder/migration_scripts/scriptGenerator/scriptGenerator_Executable/config/uim-db-connection.yaml**:

   ```
   javax:
     sql:
       DataSource:
         uim:
           connectionFactoryClassName: oracle.jdbc.pool.OracleDataSource
           URL: jdbc:oracle:thin:@<HOST>:<PORT>/<SID>
           user: <UIM_SCHEMA>
           password: xxxxxx
           inactiveConnectionTimeout: 3000
   ```

5. Modify the topology JSON files in **$WORKSPACEDIR/ata-builder/migration_scripts/scriptGenerator/scriptGenerator_Executable/topologyjsonfiles/** and run the following commands:
   Approach 1: **java -jar scriptgenerator_dblink-1.0-jar-with-dependencies.jar**

   Approach 2: **java -jar scriptgenerator_localCopy-1.0-jar-with-dependencies.jar**

6. Dynamic Attribute Migration: Once the **scriptgenerator_<Approach>-1.0-jar-with-dependencies.jar** is run, the SQLs required for Dynamic attribute migration are generated in $WORKSPACEDIR/ata-builder/migration_scripts/scriptGenerator/scriptGenerator_Executable/scriptOutFiles/dynamicAtt.sql. Run the SQL queries sequentially.
7. Verify the migrated data by going through tables with `%_FINAL` or `%_NEW` name.
8. Index Rebuild: The tables with names as `%_FINAL` and `%_NEW` contain the actual migrated data and indexes and constraints have to be added to these tables, these are generated in $WORKSPACEDIR/ata- builder/migration_scripts / scriptGenerator/scriptGenerator_Executable/ scriptOutFiles/indexRebuild.sql. Run the SQL queries sequentially.
9. In case of performing data migration using Approach 2, export the migrated Topology Data and import the migrated Topology Data into the target PDB where the ATA schema is expected to be.
10. Oracle Optimizer determines the cost of each execution plan based on database, schema, table and other statistics. The changes inside database result in stale statistics. To gather new statistics, run the following command:

    ```
    EXEC DBMS_STATS.gather_schema_stats( '<TopologySchema_Name>' );
    ```


Copyright © 2023, 2026, Oracle and/or its affiliates.

> **Note**
>
> PG_PROFILE tables which store the Service Topology Data are not supported in existing migration. If you want service topology profile data in the topology schema you can create a new service configuration and approve it. In 7.5.1.0.0, Profile Data is created for every service configuration in Approved State.

Customizing Topology JSON files for Migration

The $WORKSPACEDIR/ata-builder/migration_scripts/scriptGenerator/scriptGenerator_Executable/topologyjsonfiles/ contains three topology JSON files:

- topologyAttributeMapping.json
- topologyRoleMapping.json
- topologySpecificationMapping.json

Customize topologyAttributeMapping.json

```
[
    {
        "name": "LogicalDeviceDAO",
        "properties": [
            {
                "name": "NativeEMSName",
                "property": "NativeEMSName",
                "vertex": "",
                "columnName": ""
            }
        ]
    }
]
```

TopologyAttributeMapping (TAM) is an array defining how attributes of different DAO's can map to Topology Schema. Each TAM object consists of key-value pairs of **name** and **properties**.

- **name** – Maps to different entity classes and entity specification classes. For example: "LogicalDeviceDAO", "EquipmentSpecificationDAO", "PlaceSpecificationDAO", "PropertyLocationDAO" and so on.
- **properties** – This is an array defining how individual attributes of an entity are supposed to be stored in Topology schema. Each JSON object of the **properties** has:
  - name – Name of the Attribute.
  - property – Name of the key used to store the value retrieved from Attribute.
  - vertex – Build the relationship with the Vertices, from Topology Schema.
  - columnName – Column from Topology Schema used to store the Attribute values.

Copyright © 2023, 2026, Oracle and/or its affiliates.

> **Note**
>
> In “properties” array objects, “name” is a mandatory field to be provided which maps to either “property” or “vertex” or “columnName”.

An example of TAM is:

Assume, the topologyAttributeMapping.json contains the following:

```
[
    {
        "name": "LogicalDeviceSpecificationDAO",
        "properties": [
            {
                "name": "vendorName",
                "property": "",
                "vertex": "vendor",
                "columnName": ""
            },
            {
                "name": "modelnumber",
                "property": "Model",
                "vertex": "",
                "columnName": ""
            }
        ]
    },
    {
        "name": "EquipmentDAO",
        "properties": [
            {
                "name": " NativeEMSName",
                "property": "",
                "vertex": "",
                "columnName": "DEVICEIDENTIFIER"
            }
        ]
    }
]
```

In the above example:

- LogicalDeviceSpecification table from UIM schema is expected to have “vendorName” and “modelnumber” columns which are used to do the following:
  - All LogicalDeviceSpecification’s which have a vendorName as some non-null value is moved to PG_VENDOR table and containment edges between the devices of LogicalDevice type and their respective vendors are created in PG_DEVICE_TO_VENDOR table.
    Example: Assume there are 2 Logical Devices (“LDSampleDevice1” and “LDSampleDevice2”) of specification “LDSampleSpec”, and “LDSampleVendor” is the “vendorName”. Then, vertex/record for “LDSampleVendor” is created in PG_VENDOR table and the logical devices have their respective containment edges to the “LDSampleVendor” in PG_DEVICE_TO_VENDOR table.

Copyright © 2023, 2026, Oracle and/or its affiliates.

- All LogicalDeviceSpecification's which have a "modelnumber" as some non-null value is stored in "PROPERTIES" column of PG_DEVICE table. For example: "LDSampleSpec" has "APTS-123" as "modelnumber", then it's stored as:

```
{
    "Model": "APTS-123"
}
```

- Equipments which have non-null value in "NativeEMSName" are stored in "DEVICEIDENTIFIER" column of PG_DEVICE table.

Customizing "topologyRoleMapping.json"

```
[
    {
        "name": "ADM",
        "entityClass": [
            "LogicalDeviceDAO",
            "PhysicalDeviceDAO",
            "EquipmentDAO"
        ],
        "property": "",
        "vertex": "domain",
        "columnName": ""
    }
]
```

TopologyRoleMapping (TRM) is an array defining how entities which are role-enabled are stored in Topology schema. Each TRM object contains key-values pairs of "name", "entityClass", "property", "vertex" and "columnName".

- name – Name of the Role.
- entityClass – Entities which are enabled by the role and want data migrated for.
- property – Name of the key used to store the Role.
- vertex – Build the relationship with the Vertices, from Topology Schema.
- columnName – Column from Topology Schema used to store the Role.

> **Note**
>
> In each TRM object "name" is a mandatory field with role information which can be mapped to either "property" or "vertex" or "columnName". If "entityClass" is empty ([]) that is same as role information to be checked in Logical Device, Equipment, Physical Device, Place, Pipe and Network.

An example of TRM is:

Assume, the topologyRoleMapping.json contains the following:

```
[
    {
        "name": "ADM",
        "entityClass": [
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
            "LogicalDeviceDAO",
            "PhysicalDeviceDAO",
            "EquipmentDAO"
        ],
        "property": "",
        "vertex": "domain",
        "columnName": ""
    },
    {
        "name": "EIGRP",
        "entityClass": [
            "LogicalDeviceDAO"
        ],
        "property": "routingProtocol",
        "vertex": "",
        "columnName": ""
    },
    {
        "name": "Router",
        "entityClass": [
            "EquipmentDAO"
        ],
        "property": "",
        "vertex": "",
        "columnName": "nodeCategory"
    }
]
```

In the above example,

- A record for **ADM** is created in PG_DOMAIN table and all logical devices, equipments, and physical devices that are enabled by the **ADM** role, have the corresponding records in the PG_DEVICE_TO_DOMAIN table.
- All logical devices enabled by the **EIGRP** role have the **PROPERTIES** column populated with;

```
{
    "routingProtocol": "EIGRP"
}
```

- All equipments enabled by the **Router** role have **Router** stored in the **NODECATEGORY** column of PG_DEVICE table.

Customizing “topologySpecificationMapping.json”

```
[
    {
        "name": "EthernetDevice",
        "entityType": "LogicalDeviceSpecificationDAO",
        "relatedVertices": [
            {
                "vertex": "domain",
                "value": "Ethernet"
            }
        ],
        "characteristics": [
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
            {
                "name": "zoneID",
                "property": "",
                "vertex": "",
                "columnName": "ZONEID"
            }
        ]
    }
]
```

TopologySpecificationMapping (TSM) is an array defining how characteristics of a specification are mapped Topology schema and how all entities of a specification can have containment edge to other entities. Each TSM object contains key-values pairs of “name”, “entityType”, “relatedVertices” and “characteristics”.

- name – Name of the Specification.
- entityType – The type of entity does the specification represent.
- relatedVertices – Create containment edges for all entities of the given specification with the vertex and value. This contains an array of objects which have:
  - vertex – To which vertex the containment edges must be created to.
  - Value – The value of the vertex.
- characteristics – Array of characteristics provided by the specification and how they are stored in Topology schema.
  - name – Name of the characteristic (case-sensitive).
  - property- Name of the key used to store the characteristic.
  - vertex – Build the relationship with vertices in Topology schema.
  - columnName – Column from Topology schema in which the characteristic is stored.

> **Note**
>
> In each TSM object “name” and “entityType” are mandatory fields with specification and type of specification information. “relatedVertices” is used to create direct containment edges for all entities of the specification in question. “characteristics” is an array of objects where “name” is mandatory and talks about the characteristics provided by specification and can be mapped to either “property” or “vertex” or “columnName”.

An example of TSM is:

Assume, the topologySpecificationMapping.json contains the following:

```
[
    {
        "name": "cableModem",
        "entityType": "PhysicalDeviceSpecificationDAO",
        "characteristics": [
            {
                "name": "deviceType",
                "property": "deviceType",
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
                "vertex": "",
                "columnName": ""
            }
        ]
    },
    {
        "name": "EthernetDevice",
        "entityType": "LogicalDeviceSpecificationDAO",
        "relatedVertices": [
            {
                "vertex": "domain",
                "value": "Ethernet"
            }
        ],
        "characteristics": [
            {
                "name": "Tech",
                "property": "",
                "vertex": "Technology",
                "columnName": ""
            }
        ]
    },
    {
        "name": "Generic_Address",
        "entityType": "PlaceSpecificationDAO",
        "characteristics": [
            {
                "name": "CityName",
                "property": "",
                "vertex": "",
                "columnName": "city"
            },
            {
                "name": "StateName",
                "property": "",
                "vertex": "",
                "columnName": "state"
            },
            {
                "name": "PostalCode",
                "property": "",
                "vertex": "",
                "columnName": "postalCode"
            }
        ]
    }
]
```

In the above example,

Copyright © 2023, 2026, Oracle and/or its affiliates.

- “cableModem” is a PhysicalDeviceSpecification which has a characteristic “deviceType”. This characteristic is stored in “PROPERTIES” column of PG_DEVICE table.

```
{
    "DeviceType": "deviceType"
}
```

- A record for “Ethernet” is added to PG_DOMAIN table. All devices of “EthernetDevice” specification have containment edges to “Ethernet” in PG_Device_To_Domain table.
- “EthernetDevice” has a characteristic called “Tech”, so all unique values of “Tech” characteristic are added to PG_Technology. And for each “EthernetDevice” depending on its “Tech” characteristic respective containment edges are built.
- “Generic_Address” is a Place which has “CityName”, “StateName” and “PostalCode” characteristics which are mapped to “CITY”,”STATE” and “POSTALCODE” columns of PG_LOCATION table.

Customizing Topology JSON Files

To customize the topology JSON files:

1. When migrating Attribute or Role or Characteristic data to “PROPERTIES” column of respective entity, make sure the key used doesn’t include any empty space or special characters:

```
{
    "name": "Vendor Name",
    "property": "",
    "vertex": "vendor",
    "columnName": ""
}
```

   The above example “Vendor Name” contains empty space. Instead use “VendorName” or “Vendor_Name”.

2. In topologySpecificationMapping.json if the characteristic being migrated has length greater than 30 characters or contains special characters, the <ENTITY>_CHAR_MIG, do not have the characteristic as is. Instead, it has been casted to coded value, which can be derived from “CHARACTERISTICS_TABLE_MAPPING_MIG” in UIM schema.

   For example: “Inter-rack_Power_Distribution” (CHAR_NAME) is the name of the characteristic which has been casted to “C46575002” (COLUMN_NAME).

```
{
    "name": "Inter-rack_Power_Distribution",
    "property": "",
    "vertex": "",
    "columnName": "nodeCategory"
}
```

The above example would result in a column not found error, instead characteristic must be migrated as follows:

```
{
    "name": "C46575002",
    "property": "",
    "vertex": "",
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
    "columnName": "nodeCategory"
}
```

Topology Migration Verification Scripts

Topology verification scripts can be used only when you use Approach 2 in the ATA migration process. These scripts are present in the **$WORKSPACEDIR/ata-builder/migration_scripts/migration_verification** directory.

After the ATA migration is complete, run the scripts in the following order to validate that all entries have been migrated successfully:

1. `01_SELECT_GRANT.sql`
   Run this script in the PDB admin schema to grant read permissions on UIM tables to the topology user. Set the **UIMDATA** variable to the UIM schema user name and the **TOPDATA** variable to the topology schema user name.
2. `02_SYNONYM_CREATION.sql`
   Run this script in the ATA schema. Set the **UIMDATA** variable to the UIM schema user name.
3. `03_TOPOLOGY_VERIFICATION_SCRIPT.sql`
   After migration is complete, run this script on the topology database to verify the entity counts for **LOGICAL DEVICE**, **PHYSICAL DEVICE**, **EQUIPMENT**, **LOCATION**, **NEC**, **PIPE**, **CONNECTIVITY**, and **NETWORK** entities.

Scripts for Encrypting Existing ATA Characteristic Attributes

The process of encrypting the existing ATA characteristic attributes is similar to the existing dynamic data migration approach that includes:

1. Providing appropriate mapping JSON files and running the selected approach script generator JAR file.
2. Running the dynamically generated SQL scripts connected to ATA schema.

Assumptions:

- The `UIM_ENCRYPT` function is available in the UIM schema.

  > **Note**
  >
  > If the function is not available, see *UIM Developer's Guide* for information on running the load encryption functions.

- Only **TopologySpecificationMapping.json** is being used, and the JSON file is valid and properly formatted. See "Customizing Topology JSON files for Migration" for more information.

Approaches

Here are the approaches that you can follow, which are similar to ATA dynamic attribute migration:

- Approach 1: Using the DB link.
- Approach 2: Using the local copy.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Approach 1: Using the DB Link

Prerequisites: The DB Link should be created between ATA and UIM schema. Run the following commands from ATA schema by providing the details of UIM schema to create a DB Link:

```
DROP DATABASE LINK rem_schema;

CREATE DATABASE LINK rem_schema CONNECT TO <UIM_Schema> IDENTIFIED BY
<UIM_Schema_Password> USING '(DESCRIPTION= (ADDRESS=(PROTOCOL=TCP)
(HOST=<HOSTNAME>)(PORT=<PORT>)) (CONNECT_DATA=(SERVICE_NAME=<SID>)) )';
```

Process

To use the DB link:

1. Place the **TopologySpecificationMapping.json** file in the appropriate directory along with the JAR file.
2. Enter the following UIM schema details in the **uim-db-connection.yaml** file. You find the file at **$WORKSPACEDIR/ata-builder/migration_scripts/scriptGenerator/scriptGenerator_Executable/config**.

   ```
   javax:
     sql:
       DataSource:
         uim:
           connectionFactoryClassName: oracle.jdbc.pool.OracleDataSource
           URL: jdbc:oracle:thin:@<HOST>:<PORT>/<SID>
           user: <UIM_SCHEMA>
           password: xxxxxx
           inactiveConnectionTimeout: 3000
   ```

3. Run the JAR file in encrypt mode as follows:

   ```
   java -jar scriptgenerator_dblink-1.0-jar-with-dependencies.jar --mode
   encrypt
   ```

4. Run the **dynamicAttColEncryptAlter.sql** script before running **dynamicAttEncrypt.sql** when enabling the encryption feature. This script increases the column size to accommodate the larger encrypted data for characteristics:
5. Locate the generated SQL script named **dynamicAttEncrypt.sql** in the **outputFiles** folder.
6. Run the SQL script connected to the ATA schema.
7. Verify that the data is encrypted as expected.

Approach 2: Using the Local Copy

Prerequisites: Use this approach when both schemas are in the same PDB. Run the following commands as a database administrator to grant access to the `UIM_ENCRYPT` function and create a synonym for the ATA schema:

```
-- Grant EXECUTE on the function
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
GRANT EXECUTE ON <UIM_SCHEMA>.UIM_ENCRYPT TO <ATA_SCHEMA>;

-- Create synonym in ATA schema

CREATE OR REPLACE SYNONYM <ATA_SCHEMA>.UIM_ENCRYPT FOR
<UIM_SCHEMA>.UIM_ENCRYPT;
```

Process

To use the local copy:

1. Place the **TopologySpecificationMapping.json** file in the appropriate directory along with the JAR file.
2. Enter the following UIM schema details in the **uim-db-connection.yaml** file. You find the file at **$WORKSPACEDIR/ata-builder/migration_scripts/scriptGenerator/scriptGenerator_Executable/config**.

   ```
   javax:
     sql:
       DataSource:
         uim:
           connectionFactoryClassName: oracle.jdbc.pool.OracleDataSource
           URL: jdbc:oracle:thin:@<HOST>:<PORT>/<SID>
           user: <UIM_SCHEMA>
           password: xxxxxx
           inactiveConnectionTimeout: 3000
   ```

3. Run the JAR file in encrypt mode as follows:

   ```
   java -jar scriptgenerator_localCopy-1.0-jar-with-dependencies.jar --mode
   encrypt
   ```

4. Run the **dynamicAttColEncryptAlter.sql** script before running **dynamicAttEncrypt.sql** when enabling the encryption feature. This script increases the column size to accommodate the larger encrypted data for characteristics.
5. Locate the generated SQL script named **dynamicAttEncrypt.sql** in the **outputFiles** folder.
6. Run the SQL script connected to the ATA schema.
7. Verify that the data is encrypted as expected.

Copyright © 2023, 2026, Oracle and/or its affiliates.

14

Data Migration and Dynamic Attribute Mapping between UIM and SmartSearch

This chapter describes how to perform data migration and Dynamic Attribute mapping from UIM to SmartSearch or OpenSearch NoSQL Database. This migration helps you to seamlessly integrate data between UIM and SmartSearch or OpenSearch while designing networks.

Prerequisites

Before you perform SmartSearch data migration, ensure the following are performed:

- Cancel or complete all BIs that are associated with devices, connectivities, and pipes.
- Upgrade UIM.
  - For UIM Cloud Native, see UIM Cloud Native Upgrade Procedures
  - For Traditional UIM, see Upgrading Unified Inventory Management
- Ensure that SmartSearch and OpenSearch are deployed and are running with **Authentication** `disabled`. For more information, see "Deploying OpenSearch and OpenSearch Dashboard" and "Deploying SmartSearch".
- Add **lang_pipeline** in OpenSearch. For more information, see "Creating Ingest Pipeline for OpenSearch".
- Create all required indexes or metadata in OpenSearch. For more information, see "Installing OpenSearch" and "Creating SmartSearch Index and Metadata".
- Update the **hostAliases** section according to your SmartSearch and OpenSearch service setup. Add the target machine details in the **/etc/host** file as follows:

  ```
  <IP ADDRESS>  <instance>.<project>.<hostSuffix>
  t3.<instance>.<project>.<hostSuffix>
  admin.<instance>.<project>.<hostSuffix>
  <instance>.<project>.topology.<hostSuffix>
  ```

- Verify your access to the service URL and database connection as follows:
  1. Open a command line window and login to SQL*Plus for the UIM database.
  2. Verify if UIM schema connection is successful using SQL*Plus.
  3. Ensure that you have admin or system access for UIM schema.
  4. Open a web browser and verify if you can access the SmartSearch API URL: **https://** <instance>.<project>.topology.<hostSuffix>**:<PORT>/<API-VERSION>/index/smartsearch-location**.
  5. Verify if **JAVA_HOME** and **PATH** are set.
  6. If SmartSearch is SSL enabled, use the **Import SSL** certificate in JAVA to secure connection.

Copyright © 2023, 2026, Oracle and/or its affiliates.

7. Download the corresponding **.pem** file from target machine and run the following command in terminal to import certificate:

```
keytool -import -alias common-cert -file commoncert.pem -storetype JKS -
keystore $JAVA_HOME/jre/lib/security/cacerts
```

Running the SmartSearch Migration Script

When you perform ATA migration along with SmartSearch migration, and if you perform data migration from UIM to ATA, you can ignore running the SQL files from migration SQL script as MIG tables are already present in your UIM schema.

If ATA migration is done before you start SmartSearch migration, you must drop all the tables before running the scripts.

If you perform data migration from UIM to SmartSearch, run the migration script as follows:

1. Go to **$WORKSPACEDIR/ata-builder/migration_scripts/Char_Mig_tables** directory. You can see the following SQL files:
   a. CREATE_CHAR_MIG_TABLE.sql
   b. MIGRATION_CHAR1.sql
   c. MIGRATION_CHAR2.sql
   d. MIGRATION_CHAR3.sql
2. Run the SQL scripts in UIM schema in the above order.
3. Make a new directory:

```
mkdir $WORKSPACEDIR/<directory_name>
```

4. Copy $WORKSPACEDIR/**uim-image-builder/staging/cnsdk/uim-db/staging/uim-db-installer/ora_uim_dbtools.jar** into the newly created directory.
5. Extract the jar file as follows:

```
jar -xvf ora_uim_dbtools.jar
```

6. Modify **smartSearchMappings.json** for dynamic attribute mapping and save it.

> **Note**
>
> Dynamic attribute mappings support mapping of vendor-specific characteristic for Physical Device. You can map only one characteristic from the Physical Device entity to the **vendor** field in the OpenSearch index, using the mappings. The equivalent vendor characteristic mappings must be present in **topologySpecificationMapping.json**.

7. Modify the required parameters in the **smartSearchMigration.sh file**. For more information, see "Post-migration Check".
8. Grant the **execute** permissions to the **smartSearchMigration.sh** file. If you do not have modify permissions, update the required parameters and save the file as follows:

```
chmod +x smartSearchMigration.sh
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

9. Run the migration command using:

```
./smartSearchMigration.sh UIMToSmartSearch
```

> **Note**
>
> You must wait until the migration script is complete. Or, if you want more options, run `./smartSearchMigration.sh help`.

If the migration script fails, and you want to perform the migration again, delete the **temp** tables as follows:

```
./smartSearchMigration.sh deleteTable force
```

After a table is deleted, run the following command to restart the migration:

```
./smartSearchMigration.sh UIMToSmartSearch
```

Post-migration Check

Perform the following after a successful data migration to SmartSearch:

1. Verify or search the following information in the console log:

```
**** After Json Data Processed Completed *****
          Total Batch Size: 8
          Total Completed Batch Size: 8
          Total Pending And Failed Batch Size: 0
   **** Data Migration from UIM to Open Search Successfully Completed!
*****
```

> **Note**
>
> The **Total Pending and Failed Batch Size** value is Zero if the data migration is successful.

2. Contact your administrator to enable Authentication for SmartSearch API and OpenSearch database.
3. After enabling Authentication, open NPD to verify the search functionality for the corresponding entities such as logical device, physical device, equipment, and so on.

The sample file is as follows:

```
[
  {
    "name": "Ciena 3928 PD",
    "entityType": "PhysicalDeviceSpecificationDAO",
    "characteristics": [
      {
        "name": "vendor",
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
        "smartSearchName": "vendor"
      }
    ]
  },
  {
    "name": "Nokia 1800 PD",
    "entityType": "PhysicalDeviceSpecificationDAO",
    "characteristics": [
      {
        "name": "vendor",
        "smartSearchName": "vendor"
      }
    ]
  },
  {
    "name": "Ciena 3928 PD",
    "entityType": "PhysicalDeviceSpecificationDAO",
    "relatedVertices": [
      {
        "vertex": "vendor",
        "value": "Ciena",
        "smartSearchName": "vendor"
      }
    ],
    "characteristics": []
  }
]
```

The **relatedVertices** and **characteristics** sections in the mapping are mutually exclusive. Meaning, you should use only one of them, not both. If both sections are provided, the **characteristics** section overrides the **relatedVertices** section.

The **relatedVertices** section specifies a default value for a characteristic in the mapping, than searching for it from the database. For example, in the above sample, physical devices with the specification **Ciena 3928 PD** always have **Ciena** as the value in the vendor field in the index. This ensures that all physical devices with the **Ciena 3928 PD** specification have **Ciena** as their vendor in the NPD search.

Table 14-1 Dynamic Attributes

Dynamic Attribute	Description
name	Name of the entity specification on which the characteristic to be mapped exists. This is a mandatory attribute.
entityType	Entity specification DAO. This is a mandatory attribute. The only valid value is **PhysicalDeviceSpecificationDAO**.
characteristics.name	Name of the characteristic that you plan to map with the index attribute.
characteristics.smartSearchName	Attribute name of the index where dynamic attribute value is stored in OpenSearch index.
relatedVertices.vertex	The **relatedVertices.vertex** value from the ATA mappings. The only supported value is **vendor**.
relatedVertices.value	Manually-entered value for characteristics that will persist in attribute of the index.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Table 14-1 (Cont.) Dynamic Attributes

Dynamic Attribute	Description
relatedVertices.smartSearchName	Attribute name of index where dynamic attribute value is stored in OpenSearch index.

Open the **smartSearchMigration.sh** file in terminal or in a text editor, modify the required attributes values, and save the file. The sample file is as follows:

```
export SMART_SEARCH_BASE_URL="";  #BASE URL format like http or https://
<host>:<port>/<api_version>
   export LANG=""; # for example en
   export TENANT_ID=""; # for example tenant1
   export LIMIT=""; #for example 10000
   export BATCH_SIZE_LIMIT=""; # for example 10
   export BULK_LOAD_TIMEOUT="20";   # Maximum processing time per batch
operation (seconds)
   export INITIAL_DELAY="1";       # Initial waiting time between batch
operations (seconds)
   export BACK_OFF_FACTOR="2.0";    # Multiplier to increase retry delay time
   export MAX_DELAY="180";          # Maximum Delay allowed between retries
of one batch (seconds)
   export JAVA_HOME="";
   export DB_HOSTNAME="";
   export DB_PORT="";
   export DB_SERVICE_NAME="";
   export DB_USER_NAME="";    # user details left as blank for production
environment once prompt then you can provide
   export DB_PASSWD=""; # password details left as blank for production
environment once prompt then you can provide
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

15

Disaster Recovery Support

A minimum of two pods is required for a service to be highly available. They should be on different worker nodes (Kubernetes can schedule the pods on different nodes using pod anti-affinity). If one node goes down, it takes out the corresponding pod, leaving the other pod(s) to handle the requests until the downed pod can be rescheduled. When a worker node goes down, the PODs running on that worker node will be rescheduled on other available worker nodes.

For DB High Availability we can use the Oracle Real Application Clusters (RAC) to run a single Oracle Database across multiple servers in order to maximize availability and enable horizontal scalability.

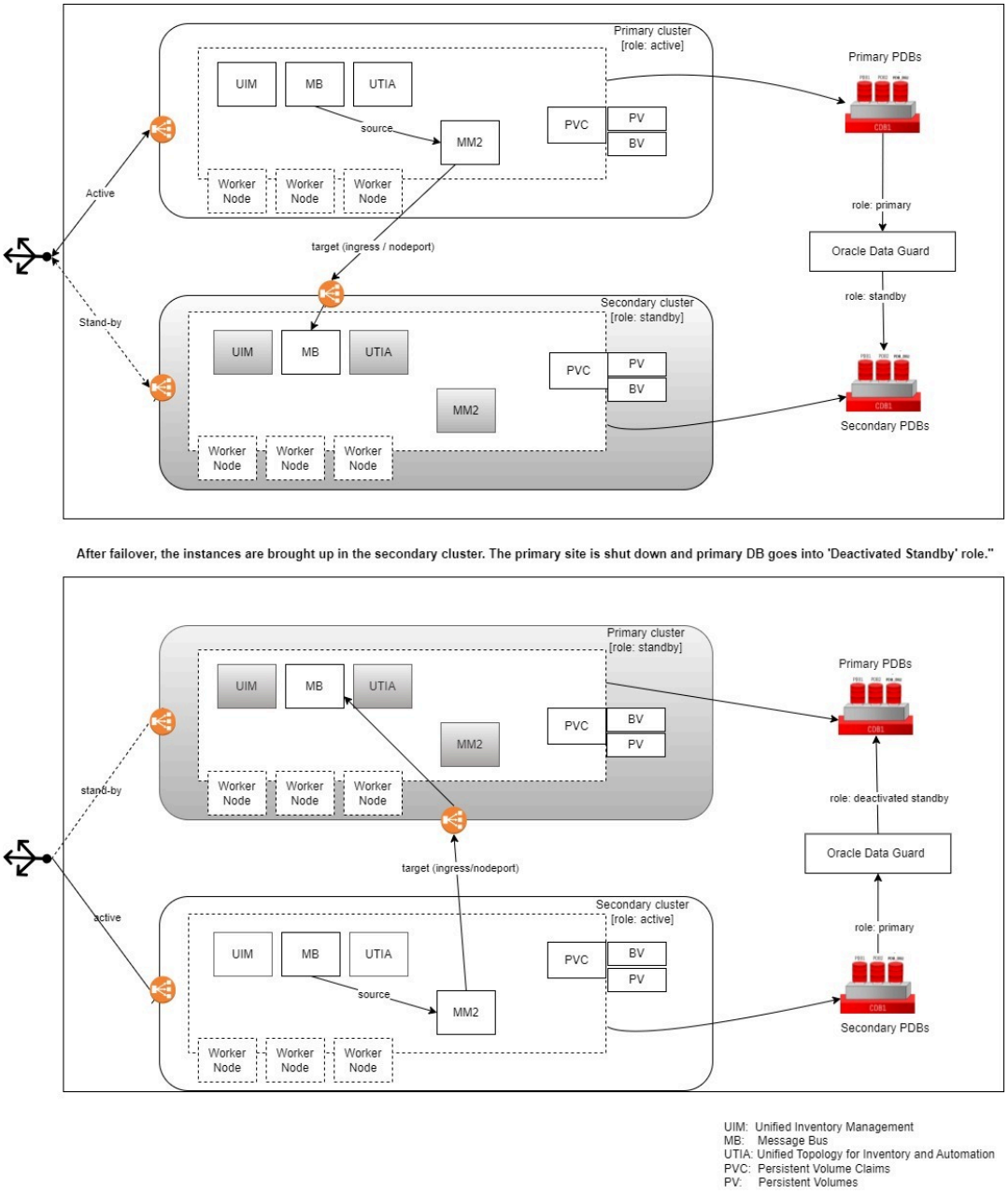
Disaster Recovery across Data Centers

The disaster recovery when the data center completely goes down is maintained with another passive data center.

Figure 15-1 documents the disaster recovery plan for the data center. A parallel passive data center is maintained, where the runtime data is periodically replicated from the active data center to the passive data center. In the event of any catastrophic failures in the primary (or active) data center, the load must be switched to secondary (or passive) data center. Before switching the load to secondary data center, you should shutdown all the services in the primary data center and start all the services in the secondary data center.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Figure 15-1 Disaster Recovery Plan for Data Center



About Switchover and Failover

The purpose of a geographically redundant deployment is to provide resiliency in the event of a complete loss of service in the primary site, due to a natural disaster or other unrecoverable failure in the primary UIM site. This resiliency is achieved by creating one or more passive standby sites that can take the load when the primary site becomes unavailable. The role reversal from the standby site to the primary site can be accomplished in any of the following ways:

- **Switchover**, in which the operator performs a controlled shutdown of the primary site before activating the standby site. This is primarily intended for planned service interruptions in the primary UIM site. Following a switchover, the former primary site

Copyright © 2023, 2026, Oracle and/or its affiliates.

becomes the standby site. The site roles of primary site and standby site can be restored by performing a second switchover operation, which is switchback.

- **Failover**, in which the primary site becomes unavailable due to unanticipated reasons and cannot be recovered. The operator then transitions the standby site to the primary role. The primary site that is down cannot act as a standby site and will require reconstruction of the database as a standby database before restoring the site roles.

About Kafka Mirror Maker

Kafka's Mirror Maker functionality makes it possible to maintain a replica of an existing Kafka cluster (which is used in Message Bus service). This mirrors a source Kafka cluster into a target (mirror) Kafka cluster. To use this mirror, it is a requirement that the source and target Kafka clusters (that is, Message Bus service) are up and running. If the target Kafka cluster is down or offline, we cannot mirror into the target cluster.

Oracle Data Guard

Oracle Data Guard is responsible for replicating transactions from the Active DB to the Standby DB. It is included as a part of every Oracle DB Enterprise Edition installation.

> **Note**
>
> When using multitenant databases involving CDBs and PDBs with Data Guard, the replication happens at the CDB level. This means all the PDBs from the active CDB will be replicated over to the standby CDB and also, the commands to enable Data Guard must be run at the CDB level.

Installation and Configuration

If ATA is disabled in UIM Cloud Native then it is not required to deploy Message Bus, ATA and Mirror Maker Services in the clusters. These commands are intended to be used as samples. For detailed documentation on deploying UIM, see "Overview of the UIM Cloud Native Deployment" in *UIM Cloud Native Deployment Guide*.

Setting up the Primary (active) Instance

To set up the primary (active) instance:

1. Provision Databases one for the primary site and another for the secondary site.
2. Set up Data Guard between primary site and secondary site. Primary site should be in **ACTIVE** role. Secondary site should be in **STANDBY** role. Refer to Oracle 19c Documentation.
3. Deploy UIM Cloud Native.
   a. Create image pull secrets (if required).
   b. Create UIM secrets for WLS admin, OPSS, WLS RTE, RCU DB and UIM DB.

Copyright © 2023, 2026, Oracle and/or its affiliates.

> **Note**
>
> `uimprimary` here refers to the Kubernetes namespace where the primary instance will be deployed. Replace this with the desired namespace.

```
$COMMON_CNTK/scripts/manage-APP-credentials.sh -p uimprimary -i dr
create wlsadmin,opssWP,wlsRTE,rcudb,uimdb -s $SPEC_PATH -a uim
```

c. Create Weblogic encrypted password.

```
$COMMON_CNTK/scripts/install-database.sh -p uimprimary -i dr -
s $SPEC_PATH -c 8 -a uim
```

d. Create UIM users secrets.

```
$COMMON_CNTK/samples/credentials/manage-app-credentials.sh -p
uimprimary -i dr -c create -f "/home/spec_dir/users.txt"
```

e. Create DB schemas.

```
$COMMON_CNTK/scripts/install-database.sh -p uimprimary -i dr -
s $SPEC_PATH -c 1 -a uim
$COMMON_CNTK/scripts/install-database.sh -p uimprimary -i dr -
s $SPEC_PATH -c 2 -a uim
```

f. Create UIM instance.

```
$COMMON_CNTK/scripts/create-ingress.sh -p uimprimary -i dr -s $SPEC_PATH
$COMMON_CNTK/scripts/create-applications.sh -p uimprimary -i dr -
s $SPEC_PATH
```

g. Add UIM user roles.

```
$COMMON_CNTK/samples/credentials/assign-role.sh -p uimprimary -i dr -f
uim-users-roles.txt
```

4. Deploy Message Bus.

```
$COMMON_CNTK/scripts/create-applications.sh -p uimprimary -i dr -
s $SPEC_PATH -a messaging-bus
```

5. Deploy ATA:

a. Create Topology DB secrets:

```
$COMMON_CNTK/scripts/manage-app-credentials.sh -p uimprimary -i dr -
s $SPEC_PATH -a ata create database
```

b. Create Topology UIM secrets:

```
$COMMON_CNTK/scripts/manage-app-credentials.sh -p uimprimary -i dr -
s $SPEC_PATH -a ata create uim
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

c. Create DB schemas:

```
$COMMON_CNTK/scripts/install-database.sh -p uimprimary -i dr -
s $SPEC_PATH -a ata -c 1
```

d. Deploy Topology:

```
$COMMON_CNTK/scripts/create-applications.sh -p uimprimary -i dr -
s $SPEC_PATH -a ata
```

See "Deploying Unified Operations Message Bus" for deploying Message Bus, "Deploying the Active Topology Automation Service" for deploying ATA.

See "Overview of the UIM Cloud Native Deployment" in *UIM Cloud Native Deployment Guide* for deploying UIM.

Setting up the Secondary (standby) Instance

To set up the secondary (standby) instance:

1. Perform switchover operation on active (primary site) DB. Now secondary site DB should be in **ACTIVE** role and primary site DB should be in **PASSIVE** role. Refer to Oracle 19c Documentation.
2. Deploy UIM Cloud Native:
   a. Export OPSS wallet file secret from primary instance and recreate in secondary instance.

> **Note**
>
> Where, `uimsecondary` refers to the Kubernetes namespace where the secondary instance will be deployed. Replace this with the desired namespace.

```
kubectl -n uimprimary get configmap uimprimary-dr-weblogic-domain-
introspect-cm -o jsonpath='{.data.ewallet\.p12}' > ./primary_ewallet.p12
$COMMON_CNTK/scripts/manage-app-credentials.sh -p uimsecondary -i dr
create opssWF -a uim
```

   b. (Optional) Create image pull secrets.
   c. Create UIM secrets for WLS admin, OPSS, WLS RTE, RCU DB and UIM DB:

```
$COMMON_CNTK/scripts/manage-app-credentials.sh -p uimsecondary -i quick
create wlsadmin,opssWP,wlsRTE,rcudb,uimdb -s $SPEC_PATH -a uim
```

   d. Create Weblogic encrypted password:

```
$COMMON_CNTK/scripts/install-database.sh -p uimsecondary -I dr -
s $SPEC_PATH -c 8 -a uim
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

e. Create UIM users secrets:

```
$COMMON_CNTK/samples/credentials/manage-uim-credentials.sh -p
uimsecondary -i dr -c create -f "/home/spec_dir/users.txt"
```

f. Create UIM instance:

```
$COMMON_CNTK/scripts/create-ingress.sh -p uimsecondary -i dr -
s $SPEC_PATH -a uim
$COMMON_CNTK/scripts/create-applications.sh -p uimsecondary -i dr -
s $SPEC_PATH -a uim
```

g. Add UIM user roles:

```
$COMMON_CNTK/samples/credentials/assign-role.sh -p uimsecondary -i dr -
f uim-users-roles.txt
```

3. Deploy message bus:

```
$COMMON_CNTK/scripts/create-applications.sh -p uimsecondary -i dr -
s $SPEC_PATH -a messaging-bus
```

4. Deploy ATA:

a. Create Topology DB secrets:

```
$COMMON_CNTK/scripts/manage-app-credentials.sh -p uimsecondary -i dr -
s $SPEC_PATH -a ata create database
```

b. Create Topology UIM secrets:

```
$COMMON_CNTK/scripts/manage-app-credentials.sh -p uimsecondary -i dr -
s $SPEC_PATH -a ata create uim
```

c. Deploy Topology:

```
$COMMON_CNTK/scripts/create-applications.sh -p uimsecondary -i dr -
s $SPEC_PATH -a ata
```

5. Deploy Mirror Maker. See "Installing and Configuring Mirror Maker 2.0" for more information.
6. After the secondary instance has been setup, switchover back to the primary (active) site.

Switchover Sequence

To perform a switchover between site A (active) and site B (standby):

1. Bring down instances in site A. These include UIM and ATA. Message Bus must be enabled to perform the replication using Mirror Maker.

```
#Disable topology
$COMMON_CNTK/scripts/delete-applications.sh -p uimprimary -i dr -
s $SPEC_PATH -a ata
#Disable UIM
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
$COMMON_CNTK/scripts/delete-applications.sh -p uimprimary -i dr -
s $SPEC_PATH -a uim
```

2. Perform switchover on DB. Site B DB will now become Primary. Site B DB will assume Standby role. Refer to Oracle 19c Documentation.
3. Bring up instances in site B. This includes UIM and ATA. Message Bus should already be active:

```
#EnableUIM
$COMMON_CNTK/scripts/create-applications.sh -p uimsecondary -i dr -
s $SPEC_PATH -a uim
#Enable topology
$COMMON_CNTK/scripts/create-applications.sh -p uimsecondary -i dr -
s $SPEC_PATH -a ata
```

4. Perform DNS switching to route all traffic to site B.

Failover Sequence

In case of any irrecoverable failure in the primary site, perform a failover operation on the standby site. To do so:

1. Perform failover on DB. Standby (secondary) DB will now become Primary. Primary site DB will assume Deactivated Standby role. Refer to Oracle 19c Documentation.
2. Bring up instances in standby. This includes UIM and Topology. Message Bus should already be active:

```
#EnableUIM
$COMMON_CNTK/scripts/create-instance.sh -p uimsecondary -i dr -
s $SPEC_PATH -a uim
#Enable topology
$COMMON_CNTK/scripts/create-applications.sh -p uimsecondary -i dr -
s $SPEC_PATH -a ata
```

3. Perform DNS switching to route all traffic to secondary instances.

Once the primary site to restored, establish a synchronization between secondary and primary site. To do so:

1. Bring up Message Bus and DB in primary site:

```
#Enable message bus
$COMMON_CNTK/scripts/create-applications.sh -p uimprimary-i dr -
s $SPEC_PATH -a messaging-bus
```

2. Setup Kafka Mirror Maker with secondary Message Bus as source and primary Message Bus as target. See "About Kafka Mirror Maker" for more information.
3. Switch primary DB role from **Deactivated Standby → Standby**. See "Deploying Unified Operations Message Bus" for more information.

As the synchronization between secondary and primary site is established, perform a switchover to the primary site. To do so:

Copyright © 2023, 2026, Oracle and/or its affiliates.

1. Bring up UIM in primary site:

```
$COMMON_CNTK/scripts/create-applications.sh -p uimprimary -i dr -
s $SPEC_PATH -a uim
```

2. Bring up Topology in primary site:

```
$COMMON_CNTK/scripts/create-applications.sh -p uimprimary-i dr -
s $SPEC_PATH -a ata
```

3. Perform DNS switching to route all traffic to primary instances.
4. Bring down instances in secondary site. This includes UIM and Topology. Message Bus should remain active for Kafka Mirror Maker synchronization:

```
#Disable topology
$COMMON_CNTK/scripts/delete-applications.sh -p uimsecondary-i dr -
s $SPEC_PATH -a ata
#Disable UIM
$COMMON_CNTK/scripts/delete-applications.sh -p uimsecondary -i dr -
s $SPEC_PATH -a uim
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

16

Checklists for Integration of Services

This chapter provides a checklist of integrating the services.

The checklists include the following variables:

- <topology-project>: Refers to the Kubernetes namespace on which the ATA service is running.
- <topology-instance>: Refers to the instance name of the ATA service running on<topology-project> namespace.
- <messaging-project>: Refers to the Kubernetes namespace on which Message Bus service is running.
- <messaging-instance>: Refers to the instance name of Message Bus running on <messaging-project> namespace.
- <loadbalancerport>: Refers to the port of loadbalancer configured. If you use Oracle Cloud Infrastructure LBaaS, or any other external load balancer, if TLS is enabled set loadbalancerport to 443. Otherwise, set loadbalancerport to 80. If there is no external loadbalancer configured for the instance, change the value of loadbalancerport to the Ingress Controller NodePort. By default if TLS is enabled on ATA HAProxy NodePort is 30543 and if TLS is disabled, is 30505.
- <loadbalancerhost>: Refers to the host of loadbalancer configured. If you use Oracle Cloud Infrastructure LBaaS, or any other external load balancer, update the value for loadbalancerhost appropriately. If there is no external loadbalancer configured for the instance change the value of loadbalancerhost to the worker node IP/ Kubernetes cluster IP.
- <hostSuffix> : Refers to the host suffix configured using **applications-base.yaml** file. The default is: uim.org.
- <oauth-token-endpoint-uri>: Get the OAuth token endpoint URI from your IdP. Usually, you can find it on **.well-known/openid-configuration** endpoint of your IdP.
- <oauth-scope>: Provide the configured scope to your OAuth client. If not configured, keep it empty.
- <oauth-audience>: Provide the configured audience to your OAuth client. If not configured, keep it empty.
- <oauth-client-id>: Provide the Client ID of of your OAUTH 2.0 client.
- <oauth-client-secret>: Provide the Client Secret of your OAUTH 2.0 client.
- <smartsearch-instance>: Refers to the instance name of the Smart Search service on <smartsearch-project> namespace.

> **Note**
>
> It is mandatory to deploy all applications with the same instance and project names with Message Bus as an exception as it can be in a different instance or project.

Copyright © 2023, 2026, Oracle and/or its affiliates.

.

Use the following checklist for integrating UIM cloud native instance, Message Bus, and ATA:

Table 16-1 Checklist for UIM cloud native instance, Message Bus, ATA, Smart Search, and Authorization Service

Source Application	Target	Application Properties	Configuration Reference

UIM CN	ATA	**ATA API**
		
		`disableTopology=false`
		`microServiceEnabled=true`
		
		`microServiceUrl=http://<topology-project>-<topology-instance>-ata-api.<topology-project>.svc.cluster.local:8080/topology/v2/`
		
		ATA UI
		
		`uim.rest.filter.CORSAllowedOrigin=https://<topology-instance>.<topology-project>.topology.<hostSuffix>:<loadbalancerport>`
		
		`topology.ui.host=https://<topology-instance>.<topology-project>.topology.<hostSuffix>`
		`topology.ui.port=<loadbalancerport>`
		`topology.ui.path=/apps/ata-ui`
			Rename custom-config.properties.samples file and update the properties.
			`mv $SPEC_PATH/project/instance/config/uim/system-config.properties.samples $SPEC_PATH/project/instance/config/uim/system-config.properties`

	Message Bus	`bootstrap.server.url=<messaging-project>-<messaging-instance>-messaging-kafka-bootstrap.<messaging-project>.svc.cluster.local:9092`	

Copyright © 2023, 2026, Oracle and/or its affiliates.

Table 16-1 (Cont.) Checklist for UIM cloud native instance, Message Bus, ATA, Smart Search, and Authorization Service

Source Application	Target	Application Properties	Configuration Reference

	Smart Search	#Uncomment the following properties only if UIM CN and Smart Search are not in the same namespace.
		
		`smartSearch.ui.host=http://<smartsearch-project>-<smartsearch-instance>-smart-search.<smartsearch-project>.svc.cluster.local`
		
		`smartSearch.ui.port=8080`	

	IDP OAUTH CLIENT	`oauth.enabled=true`
		`oauth.token.endpoint.uri=<oauth-token-endpoint-uri>`
		`oauth.client.id=<oauth-client-id>`
		`oauth.client.secret=<oauth-client-secret>`
		`oauth.scope=<oauth-scope>`
		`oauth.audience=<oauth-audience>`	

	IDP SAML 2.0		See **Enabling SAML Based Authentication Provider** in Cloud Native Deployment Guide.

	SSL	Enable ssl flag in **applications-base.yaml**
		
		`ssl:`
		`  enabled: true`
			See **Setting Up UIM Cloud Native for Incoming Access** in Cloud Native Deployment Guide.

ATA	UIM		See **Creating Secrets for UIM Credentials**

Copyright © 2023, 2026, Oracle and/or its affiliates.

Table 16-1 (Cont.) Checklist for UIM cloud native instance, Message Bus, ATA, Smart Search, and Authorization Service

Source Application	Target	Application Properties	Configuration Reference

	Message Bus	`messagingBusConfig:`
		`  namespace: <messaging-bus-namespace>`
		`  instance: <messaging-bus-instance>`
			`$SPEC_PATH/$PROJECT/$INSTANCE/applications-base.yaml`

	Authorization Service	`authorizationServiceConfig:`
		`  namespace: <authorization-service-namespace>`
		`  instance: <authorization-service-instance>`
			`$SPEC_PATH/$PROJECT/$INSTANCE/applications-base.yaml`

	SSL	Enable tls flag in **applications-base.yaml**
		`tls:`
		`  enabled: true`
			See **Setting up Secure Communication using TLS** section.

ATA
MB
Smart Search
Authorization Service
	IDP OAUTH CLIENT	Enable authentication flag in **applications-base.yaml**
		`authentication:`
		`  enabled: true`
			See **Adding Common OAuth Secret and ConfigMap** in ATA Deployment Guide
			See **Common Configuration Options For all Services** in ATA Deployment Guide

Use the following checklist for integrating traditional UIM, Message Bus, and ATA:

Checklist for entries in **/etc/hosts** for integration:

- Message service

```
<loadbalancerIP> <messaging-instance>.<messaging-
project>.messaging.bootstrap.<hostSuffix>
<loadbalancerIP> <messaging-instance>.<messaging-
project>.messaging.broker0.<hostSuffix>
<loadbalancerIP> <messaging-instance>.<messaging-
project>.messaging.broker1.<hostSuffix>
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

- ATA service

```
<loadbalancerIP>  <topology-instance>.<topology-
project>.topology.<hostSuffix>
```

Table 16-2 Checklist for UIM, Message Bus, and ATA

Source Application	Target	Application Properties	Configuration Reference

UIM	ATA	**ATA API**
		
		`disableTopology=false microServiceEnabled=true`
		
		`microServiceUrl=https://<topology-instance>.<topology-project>.topology.<hostSuffix>:<loadbalancerport>/topology/v2/`
		
		ATA UI
		
		`uim.rest.filter.CORSAllowedOrigin=https://<topology-instance>.<topology-project>.topology.<hostSuffix>:<loadbalancerport>`
		
		`topology.ui.host=https://<topology-instance>.<topology-project>.topology.<hostSuffix>`
		`topology.ui.port=<loadbalancerport>`
		`topology.ui.path=/apps/ata-ui`
			Update system-config.properties file.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Table 16-2 (Cont.) Checklist for UIM, Message Bus, and ATA

Source Application	Target	Application Properties	Configuration Reference
	Message Bus	bootstrap.server.url=<messaging-instance>.<messaging-project>.messaging.bootstrap.<hostSuffix>:<loadbalancerport> kafka.client.isTLs=true	
	IDP OAUTH CLIENT	oauth.enabled=true oauth.token.endpoint.uri=<oauth-token-endpoint-uri> oauth.client.id=<oauth-client-id> oauth.client.secret=<oauth-client-secret> oauth.scope=<oauth-scope> oauth.audience=<oauth-audience>	
	IDP SAML 2.0		For enabling SSO authentication on UIM On Premise instance, see **Setting Up Unified Inventory Management for Single Sign-On Authentication** section in UIM Installation Guide.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Table 16-2 (Cont.) Checklist for UIM, Message Bus, and ATA

Source Application	Target	Application Properties	Configuration Reference

	SSL	You have to add MB, ATA, and IDP SSL certificates to the trust of UIM Managed Servers:
		
		`keytool -import -alias common-cert -keystore $JAVA_HOME/jre/lib/security/cacerts -file $COMMON_CNTK/certs/commoncert.pem`
		
		`keytool -import -alias idp-cert -keystore $JAVA_HOME/jre/lib/security/cacerts -file $COMMON_CNTK/certs/idpcertificate.pem`
			See **Configuring the SSL Policy/Certificate** section in System Administrator Guide.
			See **Enabling WebLogic SSL Port** section in UIM Installation Guide.

Message Bus	SSL	Update **applications-base.yaml**
		
		`tls:`
		`  enabled: true`
			See **Message Bus Ingress Listener** in "**Configuring Message Bus Listeners**"

ATA	UIM	Provide proxy server uri as UIM URL to the secret. In- <shape>/ata.yaml, set Replica count of impact-analysis-api and alarm consumer to 0.	See **Creating Secrets for UIM Credentials** section. See **Configuring the Application Specification Files** section.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Table 16-2 (Cont.) Checklist for UIM, Message Bus, and ATA

Source Application	Target	Application Properties	Configuration Reference

	Message Bus	Update **applications-base.yaml**
		
		`messagingBusConfig:`
		`  namespace: <messagingbus-project>`
		`  instance: <messagingbus-instance>`
			See **Integrate ATA Service with Message Bus Service** section.

	SSL	Make Sure you add UIM and IDP certificate to the truststore of oauthConfig secret.	See **Setting up Secure Communication using TLS** section.

ATA
Message Bus
	IDP
	OAUTH CLIENT
		Enable authentication flag in **applications-base.yaml**
		
		`authentication:`
		`  enabled: true`
			See **Adding Common OAuth Secret and ConfigMap** in ATA Deployment Guide
			See **Common Configuration Options For all Services** in ATA Deployment Guide

Integrating UIM with ATA and Message Bus

This section provides you with instructions to integrate UIM (traditional and cloud native) with ATA and Message Bus. The samples for IDCS Idp are packaged along with ATA.

Integrating UIM CN with Message Bus and ATA

To integrate UIM CN with Message Bus and ATA:

1. Update **$SPEC_PATH/$PROJECT/$INSTANCE/config/uim/system-config/custom-config.properties** file with the following details:
   - UIM CN to Message Bus service settings:

```
bootstrap.server.url=<messaging-project>-<messaging-instance>-messaging-
kafka-bootstrap.<messaging-project>.svc.cluster.local:9092
#Set below properties to pass Authentication service details
kafka.client.isOAuth=true
kafka.client.oauth.token.endpoint.uri=<oauth-token-endpoint-uri> (Ex.
https://idcs-df3**********f64b21.identity.pint.oc9qadev.com:443/
oauth2/v1/token)
kafka.client.oauth.client.id=<oauth-client-id> (Ex.
e6e0b2cxxxxxxxxxxxxxxx)
kafka.client.oauth.client.secret=<oauth-client-secret> (Ex. xxxx-xxxx-
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

```
xxxx-xxxx)
kafka.client.oauth.client.scope=<oauth-client-scope> (Ex. https://
quick.sr.topology.uim.org:30443/ataScope)
kafka.client.oauth.client.audience=<oauth-client-audience> (Ex. https://
quick.sr.topology.uim.org:30443/)
#Internal communications between kubernetes services is non-ssl. Set
kafka.client.isTLs to false.
kafka.client.isTLs=false
```

- UIM CN to ATA API settings:

```
disableTopology=false
microServiceEnabled=true
microServiceUrl=http://<topology-project>-<topology-instance>-ata-
api:8080/topology/v2/
```

- UIM CN to ATA UI settings:

```
uim.rest.filter.CORSAllowedOrigin=https://<topology-instance>.<topology-
project>.topology.<hostSuffix>:<loadbalancerport>
topology.ui.host=https://<topology-instance>.<topology-
project>.topology.<hostSuffix>
topology.ui.port=<loadbalancerport>
topology.ui.path=/apps/ata-ui
```

2. Create or restart the UIM CN instance as usual, after the above configurations.

Integrating Traditional UIM with Message Bus and ATA

To integrate traditional UIM with Message Bus and ATA:

1. Update the **system-config.properties** file with the following details:
   - UIM to Message Bus service settings:

```
Provide ingress bootstrap server details as UIM traditional instance is
outside of kubernetes cluster.
bootstrap.server.url=<messaging-instance>.<messaging-
project>.messaging.bootstrap.uim.org:<loadbalancerport>
#Set below properties to pass Authentication service details
kafka.client.isOAuth=true
kafka.client.oauth.token.endpoint.uri=<oauth-token-endpoint-uri>  (Ex.
https://idcs-df3**********f64b21.identity.pint.oc9qadev.com:443/
oauth2/v1/token)
kafka.client.oauth.client.id= <oauth-client-id> (Ex.
e6e0b2cxxxxxxxxxxxxxxx)
kafka.client.oauth.client.secret= <oauth-client-secret> (Ex. xxxx-xxxx-
xxxx-xxxx)
kafka.client.oauth.client.scope=<oauth-client-scope> (Ex. https://
quick.sr.topology.uim.org:30443/ataScope)
kafka.client.oauth.client.audience=<oauth-client-audience> (Ex. https://
quick.sr.topology.uim.org:30443/)
# External access is TLS enabled
kafka.client.isTLs=true
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

- UIM to ATA API settings:

```
disableTopology=false
microServiceEnabled=true
microServiceUrl=https://<topology-instance>.<topology-
project>.topology.<hostSuffix>/topology/v2/
```

- UIM to ATA UI settings:

```
uim.rest.filter.CORSAllowedOrigin=https://<topology-instance>.<topology-
project>.topology.<hostSuffix>:<loadbalancerport>
topology.ui.host=https://<topology-instance>.<topology-
project>.topology.<hostSuffix>
topology.ui.port=<loadbalancerport>
topology.ui.path=/apps/ata-ui
```

2. Add the Identity Providers certificate to **JAVA_HOME** as follows:

```
keytool -import -alias idp-cert -keystore $JAVA_HOME/jre/lib/security/
cacerts -file <idp-certificate-file>
```

3. Add the ATA certificate to **JAVA_HOME** as follows:

```
keytool -import -alias ata-cert -keystore $JAVA_HOME/jre/lib/security/
cacerts -file <ata-certificate>
```

4. Add the common certificate to **JAVA_HOME** as follows:

```
keytool -import -alias common-cert -keystore $JAVA_HOME/jre/lib/security/
cacerts -file $COMMON_CNTK/certs/commoncert.pem
```

> **Note**
>
> Make sure that ATA and Message bus are configured with **commoncert.pem**.

Copyright © 2023, 2026, Oracle and/or its affiliates.

17

Upgrading Unified Inventory and Topology

This chapter includes the details on upgrading Unified Inventory and Topology services.

Upgrading Unified Inventory and Topology Services

This chapter explains how to upgrade your existing Unified Inventory and Topology system to the latest release.

About Upgrading Unified Inventory and Topology

In this section, the release you are upgrading from is called the old release. The release you are upgrading to is called the new release.

Upgrading to a new release of UIM cloud native and the corresponding Unified Inventory and Topology services includes the following tasks:

- Planning the upgrade
- Performing the pre-upgrade tasks
- Upgrading operators
- Upgrading UIM cloud native and the corresponding Unified Inventory and Topology services

Supported Upgrade Paths

This release of UIM supports the direct upgrade path from release 7.5.0 or later to release 8.0.0.

Planning Your Upgrade

Before you plan your upgrade, do the following:

- Read *UIM Release Notes* of the corresponding releases for information about UIM and Unified Inventory and Topology services.
- Check *UIM Compatibility Matrix* of the new release for the supported tech stack versions.

Pre-Upgrade Tasks

This section provides the consolidated steps to verify while upgrading services from a previous release.

See "Unified Inventory and Topology Toolkit" to understand the required artifacts and specification files for the new release.

1. Download and extract the required artifacts, and assemble the specification files.
2. Copy the configuration values from the specification files (applications-base.yaml, database.yaml, app-uim.yaml, app-messaging-bus.yaml, app-ata.yaml, and app-

Copyright © 2023, 2026, Oracle and/or its affiliates.

smartsearch.yaml) from the old release to the new release specification location, especially values related to the load balancer, ingress controller, storage volume, imagePullSecret, UIM custom settings, and SAF configurations.

3. Copy your custom configuration data (such as custom-config.properties, topologyMapping, and so on.) and logging properties files from the old release to the new release specification location.

Pre-Upgrade Tasks for Releases 8.0.0.0.0 or Later

The pre-upgrade tasks are:

- In the UIM 8.0.0.0 release, **uim-cntk** has been merged with **common-cntk**. For details, see "Migrating UIM_CNTK to COMMON_CNTK" in the *Cloud Native Deployment Guide*.
- Updates have been made to the **COMMON_CNTK** command syntax for running scripts. Review and use the new command formats to ensure compatibility.
- The **SPEC_PATH** file structure has been modified. Verify and update any related configurations as needed. For more information, see "About the Specifications File".
- In this release, all modifications to configuration and logging properties must be performed within the **$SPEC_PATH/project/instance/config/** directory. Direct updates to files under **$COMMON_CNTK** are not permitted.
- This release includes major version upgrades to **FMW 14c** and **Java 21**. Ensure your environment is updated to these versions.

Upgrading Operator

This section covers the important steps required to upgrade Ingress Controller, WebLogic Kubernetes Operator, and Strimzi Operator to the latest compatible versions.

Ingress Controller Upgrade

To upgrade Ingress Controller, you must check the official documentation of your ingress controller. The following commands are sample steps to upgrade ingress HAProxy in place:

1. Export the variable to point your ingress HAProxy namespace installation as follows:

```
export HAPROXY_NS=haproxy
```

2. Add helm repository as follows:

```
helm repo add haproxytech https://haproxytech.github.io/helm-charts
```

3. Upgrade HAProxy as follows:

```
helm upgrade haproxy-kubernetes-ingress haproxytech/kubernetes-ingress --
namespace $HAPROXY_NS --version <compatible-version>
```

Upgrading WebLogic Kubernetes Operator (WLSKO)

To upgrade WebLogic Kubernetes Operator, check *UIM Compatibility Matrix* for the latest supported version and follow the instructions mentioned in "Upgrading WebLogic Operator" in *Cloud Native Deployment Guide*. The sample for in-place upgrade is as follows:

Copyright © 2023, 2026, Oracle and/or its affiliates.

1. Export variable to point to your WebLogic operator namespace installation:

   ```
   export WLSKO_NS=wlsko
   ```

2. Add Helm repository:

   ```
   helm repo add weblogic-operator https://oracle.github.io/weblogic-
   kubernetes-operator/charts --force-update
   ```

3. Upgrade the operator:

   ```
   helm upgrade weblogic-operator \
      weblogic-operator/weblogic-operator \
     --version <compatible-version> \
     --namespace $WLSKO_NS
   ```

For information related to WebLogic operator installation, see https://oracle.github.io/weblogic-kubernetes-operator/managing-operators/installation/#install-the-operator.

Upgrading Strimzi Operator

To Upgrade the Strimzi operator, you must download and use the corresponding latest **common-cntk** for installation of Strimzi.

See "Upgrading Strimzi Operator" for information on upgrading the strimzi operator to latest supported version.

Upgrading Message Bus

This section provides the instructions to upgrade Message Bus service to a new release.

Pre-Upgrade Tasks

To upgrade Message Bus zookeeper, the replica should be an odd number greater than 2.

Pre-Upgrade Tasks for Release 7.7.0.0 or Later

Perform Zookeeper to Kraft Migration. The steps are listed under "Message Bus KRaft Migration".

Upgrading Message Bus

To upgrade Message Bus instance, follow instructions listed under "Upgrading Message Bus" **Upgrading Message Bus** section. Run the following command with the latest artifacts:

```
$COMMON_CNTK/scripts/upgrade-applications.sh -p $PROJECT -i $INSTANCE -
s $SPEC_PATH -a messaging-bus
```

Upgrading UIM Cloud Native Environment

To upgrade UIM cloud native, see "Upgrading the UIM Cloud Native Environment" in *UIM Cloud Native Deployment Guide*.

Copyright © 2023, 2026, Oracle and/or its affiliates.

The section "UIM Cloud Native Upgrade Procedures" in *UIM Cloud Native Deployment Guide* provides the instructions for in-place upgrade.

Upgrading Authorization Service

To upgrade Authorization to the latest version:

1. Build the Authorization image using the latest **authorization-builder.tar.gz**. See "Creating Authorization Images" for more information.
2. To upgrade the Authorization Service instance, see "Upgrading the Authorization Instance" and run the following command:

```
$COMMON_CNTK/scripts/upgrade-applications.sh -p $PROJECT -i $INSTANCE -
s $SPEC_PATH -a authorization
```

Upgrading ATA

This section includes the instructions to upgrade ATA to a new release.

Pre-Upgrade Tasks for Release 7.8.0.0 or Later

The Authorization service is mandatory and should be deployed and must be running before creating or upgrading an ATA instance.

Upgrading ATA

To upgrade ATA:

1. Build ATA Image using the latest **ata-builder.tar.gz**. For information on creating the image, see "Creating ATA Images".
2. Update **app-ata.yaml, applications-base.yaml** and **database.yaml** in the **$SPEC_PATH/$PROJECT/$INSTANCE** directory with the required configurations provided as part old setup. Provide new image names and tags in **app-ata.yaml**.
3. **PDB Upgrade:** Upgrade ATA schema. For information on upgrading PDB, see "Upgrading the ATA Schema". Run the following command:

```
$COMMON_CNTK/scripts/install-database.sh -p $PROJECT -i $INSTANCE -
s $SPEC_PATH -a ata -c 3
```

4. **Application upgrade:** Upgrade the ATA instance. For information on upgrading instance, see "Upgrading the ATA Instance". Run the following command:

```
$COMMON_CNTK/scripts/upgrade-applications.sh -p $PROJECT -i $INSTANCE -
s $SPEC_PATH -a ata
```

Upgrading SmartSearch

This section includes the process for upgrading SmartSearch.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Pre-Upgrade Tasks

A compatible version of OpenSearch must be running and accessible to the SmartSearch service.

Upgrading SmartSearch

To upgrade SmartSearch:

1. Load the SmartSearch image after extracting it from the downloaded archive, and then push it to your remote repository.

   ```
   podman load -i <smart-search-latest.tar>
   podman tag localhost/smart-search:latest <container repository>/smart-search:<tag>
   ```

2. Update the **$SPEC_PATH/$PROJECT/$INSTANCE/app-smartsearch.yaml** with the new SmartSearch image.
3. Upgrade SmartSearch by running following command. For more information, see "Upgrading the SmartSearch Service".

   ```
   $COMMON_CNTK/scripts/upgrade-applications.sh -p $PROJECT -i $INSTANCE -s $SPEC_PATH -a smartsearch
   ```

4. Refresh the SmartSearch metadata. For more information, see "Creating SmartSearch Index and Metadata".

Upgrading UIM Cloud Native Using Staging Instance

This section describes the Blue-Green upgrade process for UIM cloud native using a staging instance, which leverages Oracle Data Guard. This approach minimizes production downtime and ensures that the latest version is verified in a staging environment before cutover.

In this approach:

- **The Blue Environment** represents the current production system that is available to all users.
- **The Green Environment** functions as a dedicated staging environment, where the backed-up UIM schema is upgraded and validated by attaching the new application. The UIM schema is secured and synchronized using Data Guard. All upgrade and test operations are performed in Green, ensuring no disruption to the Blue (production) environment. After the validation is complete, Green environment becomes the new production environment.

The Blue-Green upgrade involves the following phases:

- **Phase 1:** Staging and testing
- **Phase 2:** Staging update and Production cutover
- **Phase 3:** Upgrading the old Production (Blue) environment

Copyright © 2023, 2026, Oracle and/or its affiliates.

Prerequisites

Learn about prerequisites required for performing UIM cloud native upgrade using staging instance.

The prerequisites for performing the Blue-Green upgrade are:

- **Primary Production Environment (Blue environment)**: The environment with an older version of the application. UIM and ATA should be present on the same Oracle Container Database (CDB) for Data Guard replication.

  > **Note**
  >
  > – If the environment is older than UIM 7.5.1.x, perform data migration for ATA and OpenSearch.
  >
  > – If the environment is older than 7.7.0.x but newer than 7.5.1.x, perform data migration for OpenSearch. See *Unified Inventory and Topology Guide* for more information.

- **Secondary CDB (Green Environment)**: Provision a new container database (CDB) using the same configuration as the Blue production environment. Ensure that Repository Creation Utility (RCU) schemas are not replicated with Data Guard, as this may cause issues. To prevent replication, configure the RCU schemas for the Green environment in a separate CDB that does not have Data Guard replication enabled.

  > **Note**
  >
  > Starting with Oracle Database 23ai, PDB-level data replication using Data Guard is supported. See the Oracle Data Guard documentation for more details. With Oracle Database 19c and earlier, PDB-level replication is not supported. Ensure that the Blue and Green environments use the same schema names.

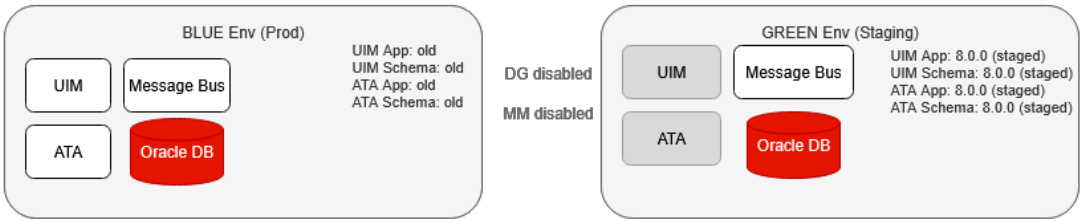
- **Establish Data Guard Configuration**: Implement Oracle Data Guard for real-time replication between Blue and Green CDBs. Set the Blue CDB to **Active** and the Green CDB to **Standby**.
- **Establish Kafka Mirrormaker Replication**: Set up Kafka MirrorMaker, configuring the Blue environment as the source and the Green environment as the target. For setting up Kafka Mirrormaker, see "Installing Mirror Maker" in *Unified Inventory and Topology Deployment Guide*.

Blue-Green Upgrade Phases

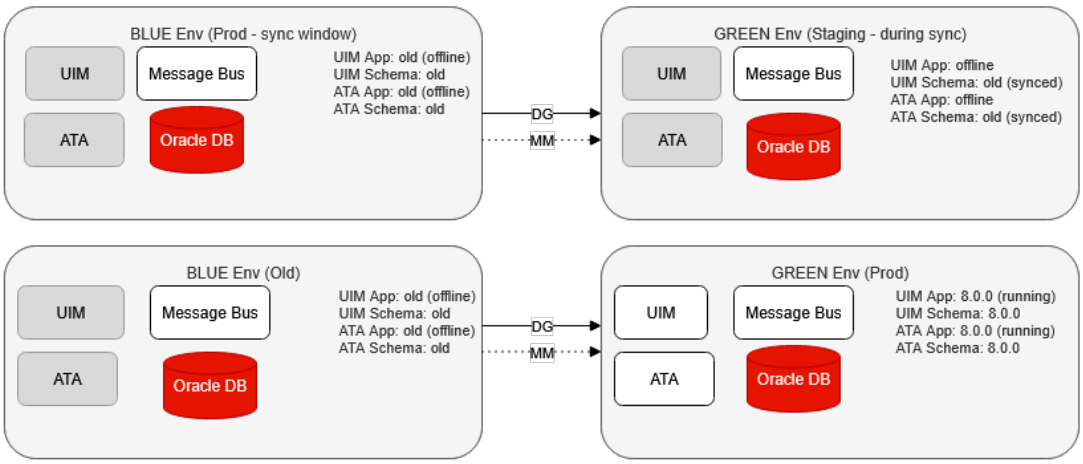
This section provides details about various phases involved in performing the Blue-Green upgrade. The following figure illustrates the various phases involved in performing Blue-Green upgrade.

Copyright © 2023, 2026, Oracle and/or its affiliates.

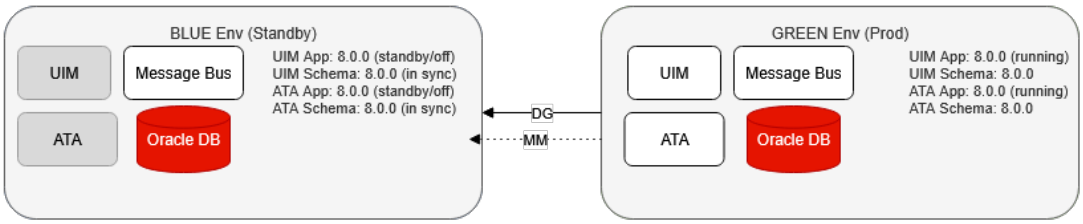
Phase 1: Staging & Testing (replication disabled)



Phase 2: Staging Update & Production Cutover (sync, upgrade, then switch)



Phase 3: Blue (Old Production) Upgrade and establish standby



Legend: Solid fill = RUNNING, Grey fill = OFFLINE; Solid line = Data Guard (DG), Dotted line = MirrorMaker (MM)

Phase 1: Staging and Testing

This section includes the tasks you perform for staging and testing.

> **Note**
>
> The Blue environment is the current live production instance and remains online throughout this phase. The Green environment is configured and upgraded separately for staging and validating with the latest application version.

In this phase:

1. Temporarily disable Data Guard replication between the Blue (source) and Green (standby) CDBs.

Copyright © 2023, 2026, Oracle and/or its affiliates.

2. Temporarily disable MirrorMaker replication from Blue (source) to Green (target) Message Bus. See "Installing Mirror Maker" in *Unified Inventory and Topology Deployment Guide* for more information.
3. Perform upgrade on the Green CDB to a database version compatible with the latest version of UIM.
4. Perform necessary tech stack upgrades to make Kubernetes, WLSKO, Strimzi, WIT, WDT, Helm, Podman, HAProxy, CRI-O compatible with the latest version of UIM. See "UIM Cloud Native Deployment Software Compatibility" in *UIM Compatibility Matrix* for the corresponding software versions.
5. Upgrade the Green environment schemas to the latest UIM version. Perform this for UIM and ATA schemas.
6. Compile all necessary cartridges using the latest SCD version and deploy in the Green environment. Perform sanity testing and validate. See "UIM Cloud Native Deployment Software Compatibility" in *UIM Compatibility Matrix* for the compatible SCD version.

Phase 2: Staging Update and Production Cutover

The production Blue environment must be made offline, resulting in a scheduled downtime.

In this phase:

1. Shutdown Blue and Green application instances (except for Message Bus). This is done by using `delete-applications.sh` script for all installed services or scaling down the replicas to **0**.
2. Reactivate Data Guard to synchronize the Green PDB with the latest data from the Blue production environment.

> **Note**
>
> This process will revert the Green environment schemas to the earlier version currently used in the production database. However, it is required to ensure that all the latest production data is transferred to the staging environment.

3. Reactivate MirrorMaker to synchronize Green and Blue Message Bus.
4. Perform ATA Migration and OpenSearch migration (if required).
5. Monitor and verify whether data synchronization is achieved.
6. Disable Data Guard and MirrorMaker after the above synchronization step.
7. Perform the schema upgrade again on the Green PDB by following the procedure mentioned in Phase 1.
8. Start the application instances in the Green environment. You can do this by running the `create-applications.sh` script for all installed services or by increasing the replica count as needed.
9. Deploy all necessary cartridges on UIM and perform the sanity test.
10. Redirect the production traffic to the upgraded Green environment.
    Once the process is complete, the Green environment will become the new production environment.

Copyright © 2023, 2026, Oracle and/or its affiliates.

Phase 3: Blue (Old Production) Upgrade

The Green environment is now the active production system. After upgrade and data synchronization, the Blue environment will serve as the backup.

In this phase:

1. Upgrade the Blue environment.
2. Initiate Data Guard replication from Green to Blue, to synchronize all data changes.
3. Initiate Mirror Maker replication from Green (source) to Blue (target) Message Bus.
4. Suspend Data Guard replication after the synchronization is complete.
5. Suspend Mirror Maker replication after the synchronization is complete.
6. Test the Blue environment to make sure everything works correctly.
7. Shut down the application instances in the Blue environment, except for the Message Bus. Retain the Blue environment as a backup.
8. Enable Data Guard replication from Green CBD to Blue CDB.
9. Enable Mirror Maker replication from Green (source) to Blue (target) Message Bus. After completing these steps, the Green environment will operate as the primary production environment, and the Blue environment will serve as the standby or backup.

Copyright © 2023, 2026, Oracle and/or its affiliates.

A
SSL Certificates

This appendix provides information on generating your SSL certificates.

The following DNs entries require a CA-signed certificate, based on the services you use for your production environment. Alternatively, you use a CA-signed certificate with wildcard for the corresponding <hostSuffix>. For example: *.uim.org.

You use self-signed certificates for a development environment.

> **Note**
>
> - DNs for Message Bus are required only when the service is accessed by Traditional UIM or by any external system such as Assurance. The number of DNS entries for Message Bus are in proportional to the number of Kafka brokers configured.
> - OpenSearch can be installed in <project> namespace or a different namespace <opensearch-namespace>. CA-Signed certificate opensearch-cluster-master.<opensearch-namespace>.svc.cluster.local is required with the namespace when OpenSearch is deployed.

Table A-1 DNS Entries for Services

DNS Entry	Service
<instance>.<project>.<hostSuffix>	UIM CN
admin.<instance>.<project>.<hostSuffix>	UIM CN
t3.<instance>.<project>.<hostSuffix>	UIM CN
<instance>.<project>.topology.<hostSuffix>	ATA
<INSTANCE>.<PROJECT>.messaging.broker0.<hostSuffix> <instance>.<project>.messaging.broker<N>.<hostSuffix> <instance>.<project>.messaging.bootstrap.<hostSuffix>	Message Bus
<instance>.<project>.opensearch.<hostSuffix>	Open Search Dashboard
opensearch-cluster-master.<opensearch-namespace>.svc.cluster.local	Open Search

Generating Self-signed Certificates

To generate self-signed certificates:

1. Create the **certs** folder under the **$COMMON_CNTK** directory.

   ```
   mkdir $COMMON_CNTK/certs
   ```


Copyright © 2023, 2026, Oracle and/or its affiliates.

2. Update the following command with the appropriate values of INSTANCE, PROJECT, and hostSuffix names and execute it to generate a common self-signed certificate and key that can be used for Message Bus, UIM, and ATA. You can add or remove the DNS entries from the below command based on requirements.

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout $COMMON_CNTK/
certs/commonkey.pem -out $COMMON_CNTK/certs/commoncert.pem -subj "/
CN=<INSTANCE>.<PROJECT>.<hostSuffix>" -extensions san -config <(echo
'[req]'; echo 'distinguished_name=req'; echo '[san]';echo
'subjectAltName=@alt_names'; \echo '[alt_names]'; \
echo 'DNS.1=<INSTANCE>.<PROJECT>.topology.<hostSuffix>'; \
echo 'DNS.2=<INSTANCE>.<PROJECT>.<hostSuffix>'; \
echo 'DNS.3=admin.<INSTANCE>.<PROJECT>.<hostSuffix>'; \
echo 'DNS.4=t3.<INSTANCE>.<PROJECT>.<hostSuffix>'; \
echo 'DNS.5=<INSTANCE>.<PROJECT>.messaging.broker0.<hostSuffix>'; \
echo 'DNS.6=<INSTANCE>.<PROJECT>.messaging.broker<N>.<hostSuffix>'; \
echo 'DNS.7=<INSTANCE>.<PROJECT>.messaging.bootstrap.<hostSuffix>'; \
echo 'DNS.8=opensearch-cluster-master.<opensearch-
namespace>.svc.cluster.local'; \
echo 'DNS.9=<INSTANCE>.<PROJECT>.opensearch.<hostSuffix>'; \
)
```

3. You can add or remove the DNS entries in the above sample certificate. Check the following scenarios for removing or adding the DNS entries:

   - If the Message Bus ingress listener is not enabled, you can remove the following DNS entries:

     ```
     quick.sr.messaging.broker0.uim.org
     quick.sr.messaging.broker1.uim.org
     quick.sr.messaging.bootstrap.uim.org
     ```

   - For traditional UIM, you can remove the following DNS entries and add the hostnames of traditional UIM servers:

     ```
     quick.sr.uim.org
     admin.quick.sr.uim.org
     t3.quick.sr.uim.org
     ```

   - If the DNS entry is for ATA: `quick.sr.topology.uim.org`
   - If the DNS entry is for OpenSearch service: `opensearch-cluster-master.sr.cluster.local`
   - If the DNS entry is for OpenSearch dashboard: `quick.sr.opensearch.uim.org`

Generating Self-Signed Wild Card SSL Certificate

To generate self-signed wild card SSL certificate:

1. Create the **certs** folder in **$COMMON_CNTK directory** as follows:

```
mkdir $COMMON_CNTK/certs
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

2. To generate a wild card SSL certificate you can update <hostSuffix> value. The default is **uim.org** and run following command:

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout $COMMON_CNTK/
certs/wildcardkey.pem -out $COMMON_CNTK/certs/wildcardcert.pem -subj "/
CN=*.<hostSuffix>" -extensions san -config <(echo '[req]'; echo
'distinguished_name=req';
echo '[san]';echo 'subjectAltName=@alt_names'; \echo '[alt_names]'; \
echo 'DNS.1=*.<hostSuffix>'; \
)
```

Note

- To use wild card certificates, you must configure **subDomainNameSeperator** field as **-**, in **applications-base.yaml** in the **spec** path location.
- WebLogic by default does not recognizes wild card certificates. In production environment, you must configure the custom hostname verifier as `weblogic.security.utils.SSLWLSWildcardHostnameVerifier`. See WebLogic documentation for setting up **hostNameVerifier**.
- In development environment, you can disable hostname verification.

Copyright © 2023, 2026, Oracle and/or its affiliates.

B

Migrating from Traefik Ingress Controller to Annotations Based Generic Ingress Controller

Prerequisites

Here are the prerequisites you need:

- Install annotation-based ingress controller.
- Delete Traefik Ingress Controller. For more information, see Setting Up Automation

Installing Generic Ingress Controller

To install generic ingress controller:

1. You can use any annotation-based ingress controller that supports standard Kubernetes ingress API. The samples for HAProxy **ingressController** are provided.
2. For installation of HAProxy, the sample values are provided under $COMMON_CNTK/samples/charts/haproxy. For more information, see Working with Ingress, Ingress Controller, and External Load Balancer

Migrating to Generic Ingress Controller for ATA

To migrate to generic ingress controller for ATA:

1. Delete Traefik Ingress Controller. For more information, see Setting Up Automation.
2. Download the latest common-cntk, assemble the specifications and retain your old, required values.
3. Make sure the value of **ingressController** in spec path is `GENERIC`.

   ```
   ingressController: "GENERIC"
   ```

4. Update the **loadbalancerport** value in **applications-base.yaml** with your **ingressController** loadbalancer or NodePort port.
5. Uncomment and provide required ingress annotations and **ingressClassName** in **applications-base.yaml** file according to your ingress controller. The samples are provided for HAProxy ingress controller.

   ```
   ingress:
     className: "haproxy"
     annotations:
        haproxy.org/cookie-persistence: "uimhaproxycookie"
   ```

6. (Optional) Provide additional annotations under **ata.ingress.annotations** tag that is specific to ATA in **app-ata.yaml**.

Copyright © 2023, 2026, Oracle and/or its affiliates.

7. Upgrade the ATA instance.

```
$COMMON_CNTK/scripts/upgrade-instance.sh -p $PROJECT -i $INSTANCE -
s $SPEC_PATH -a ata
```

8. Verify if your application is accessible through your **ingressController** port.

Migrating to Generic Ingress Controller for Message Bus

To migrate to generic ingress controller for Message Bus:

1. Download the latest common-cntk, copy the latest **applications-base.yaml** file to spec path and retain your old, required values.
2. Make sure the value of **ingressController** in spec path is `GENERIC`.

```
ingressController: "GENERIC"
```

3. If TLS is `true`, update the **loadbalancerport** value in **applications-base.yaml** with your **ingressController** SSL loadbalancer or NodePort port.
4. If TLS is `false`, update **ingressSslPort** inside the **kafka-cluster** section in **applications-base.yaml** with **ingressController** non-SSL loadbalancer or NodePort port.
5. Uncomment and provide required ingress annotations specific to Kafka cluster and **ingressClassName** in **app-messaging-bus.yaml** file according to your ingress Controller. The samples are provided for HAProxy ingress controller.

```
ingress:
  className: "haproxy"
  annotations:
     haproxy.org/cookie-persistence: "uimhaproxycookie"
```

6. For Message Bus, annotations given in **kafka-cluster** section are mandatory, under **kafka-cluster.listeners.ingress.annotations** tag in **app-messaging-bus.yaml** file.

```
kafka-cluster:
  ingress:
      ingressSslPort: 30543
      annotations:
        haproxy.org/ssl-passthrough: "true"
        ingress.kubernetes.io/ssl-passthrough: "true"
```

7. Upgrade the Message Bus instance.

```
$COMMON_CNTK/scripts/upgrade-application.sh -p $PROJECT -i $INSTANCE -
s $SPEC_PATH -a messaging-bus
```

8. Verify your application is accessible through your **ingressController** port.

Copyright © 2023, 2026, Oracle and/or its affiliates.

C

Migrating from NGINX to HAProxy Ingress Controller

This section describes the configuration changes required to migrate from the NGINX Ingress Controller to the HAProxy Ingress Controller.

> **Note**
>
> You must configure all applications to use HAProxy, and then uninstall NGINX.

Installing HAProxy Ingress Controller

To install the HAProxy Ingress Controller, see "Planning and Validating Your Cloud Environment" in *UIM Cloud Native Deployment Guide*.

Configure applications-base.yaml for All Services

The configuration in this section applies to UIM, ATA, Message Bus, and Smart Search. Perform the following before proceeding to the application-specific sections:

1. Download the latest `common-cntk`, merge the specifications, and retain the required values from your existing configuration.
2. Ensure that the `ingressController` value in the `spec` section is set to `GENERIC`:

   ```
   ingressController: "GENERIC"
   ```

3. Update the **loadbalancerport** value in the **applications-base.yaml** file with the HAProxy load balancer port or NodePort:

   ```
   loadbalancerport: 30543
   ```

4. Uncomment and configure the required ingress annotations and **ingress.className** in the **applications-base.yaml** file based on your ingress controller. The following example shows values for the HAProxy Ingress Controller:

   ```
   ingress:
     className: "haproxy"
     annotations:
        haproxy.org/cookie-persistence: "uimhaproxycookie"
   ```

Migrating UIM to the HAProxy Ingress Controller

To migrate UIM to the HAProxy Ingress Controller:

Copyright © 2023, 2026, Oracle and/or its affiliates.

1. If TLS is enabled, replace the NGINX annotations under **uim.ingress.annotations** in the **$SPEC_PATH/$PROJECT/$INSTANCE/app-uim.yaml** file.

```
uim:
  ingress:
    annotations:
      haproxy.org/ssl-redirect: "true"
      haproxy.org/backend-config-snippet: |
         http-request del-header WL-Proxy-Client-IP
         http-request del-header WL-Proxy-SSL
         http-request set-header X-Forwarded-Proto https
         http-request set-header WL-Proxy-SSL true
```

2. Recreate the UIM ingress and upgrade the instance:

```
# recreate uim ingress
$COMMON_CNTK/scripts/delete-ingress.sh -p $PROJECT -i $INSTANCE -
s $SPEC_PATH -a uim
$COMMON_CNTK/scripts/create-ingress.sh -p $PROJECT -i $INSTANCE -
s $SPEC_PATH -a uim

# upgrade uim instance
$COMMON_CNTK/scripts/upgrade-applications.sh -p $PROJECT -i $INSTANCE -
s $SPEC_PATH -a uim
```

3. Verify that the UIM instance is accessible through the HAProxy NodePort or LoadBalancer port.

Migrating ATA and SmartSearch to the HAProxy Ingress Controller

To migrate ATA and Smart Search to the HAProxy Ingress Controller.

> **Note**
>
> If the **applications-base.yaml** file is already configured, no additional configuration is required for ATA and SmartSearch.

1. Recreate the ingress and upgrade the applications:

```
# ATA Ingress
$COMMON_CNTK/scripts/delete-ingress.sh -p $PROJECT -i $INSTANCE -
s $SPEC_PATH -a ata
$COMMON_CNTK/scripts/create-ingress.sh -p $PROJECT -i $INSTANCE -
s $SPEC_PATH -a ata
$COMMON_CNTK/scripts/upgrade-applications.sh -p $PROJECT -i $INSTANCE -
s $SPEC_PATH -a ata

# Smart Search Ingress
$COMMON_CNTK/scripts/delete-ingress.sh -p $PROJECT -i $INSTANCE -
s $SPEC_PATH -a smartsearch
$COMMON_CNTK/scripts/create-ingress.sh -p $PROJECT -i $INSTANCE -
s $SPEC_PATH -a smartsearch
$COMMON_CNTK/scripts/upgrade-applications.sh -p $PROJECT -i $INSTANCE -
s $SPEC_PATH -a smartsearch
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

2. Verify that the ATA and SmartSearch instances are accessible through the HAProxy NodePort or LoadBalancer port.

Migrating OpenSearch Dashboard to the HAProxy Ingress Controller

To migrate the OpenSearch Dashboard to the HAProxy Ingress Controller:

1. Set the OpenSearch Dashboard service type to **ClusterIP**.
2. Enable ingress and update the **ingressClassName** to **haproxy** in the **$SPEC_PATH/$PROJECT/opensearch/os_board_values.yaml** file.
3. Replace the NGINX annotations with HAProxy annotations to enable session persistence:

```
service:
  type: ClusterIP

ingress:
   enabled: true
   ingressClassName: haproxy
   annotations:
      haproxy.org/cookie-persistence: "uimhaproxycookie"
```

4. Upgrade or create the OpenSearch Dashboard instance:

```
$COMMON_CNTK/samples/charts/opensearch/scripts/upgrade-opensearch-
dashboard.sh
```

5. Verify that the OpenSearch Dashboard is accessible through the HAProxy NodePort or LoadBalancer port.

Migrating Message Bus to the HAProxy Ingress Controller

To migrate the Message Bus to the HAProxy Ingress Controller:

1. Update the ingress annotations for the Kafka cluster in the **$SPEC_PATH/$PROJECT/$INSTANCE/app-messaging-bus.yaml** file to enable ingress for the Message Bus:

```
kafka-cluster:
   listeners:
     ingress:
        ingressSslPort: 30543
        annotations:
          haproxy.org/ssl-passthrough: "true"
          ingress.kubernetes.io/ssl-passthrough: "true"
```

2. Upgrade the Message Bus instance:

```
 $COMMON_CNTK/scripts/upgrade-applications.sh -p $PROJECT -i $INSTANCE -
s $SPEC_PATH -a messaging-bus
```

3. Verify that the Message Bus instance is accessible through the HAProxy NodePort or LoadBalancer port.

Cleaning Up the NGINX Ingress Controller

To remove the NGINX Ingress Controller after migrating to HAProxy:

Copyright © 2023, 2026, Oracle and/or its affiliates.

1. Verify that all applications are accessible through the HAProxy Ingress Controller.
2. Uninstall the NGINX Ingress Controller:

```
helm uninstall nginx-operator -n nginx
```


Copyright © 2023, 2026, Oracle and/or its affiliates.

D

Managing Certificate Expiry

Oracle provides utility scripts to analyze the certificates used by ATA, MB, Authorization, and SmartSearch services. You can renew the expired certificates using this script. You must follow the prerequisites and post-requisites for this script.

The guidelines for using the utility script are:

- If you are using **SSL TERMINATE** at ingress for ATA, Authorization, Message Bus, and SmartSearch services. You can run this script with appropriate arguments and renew or verify the expiry of certificates for the services one after the other or all together.
- If Ingress listener is enabled for Message Bus, you can use this script to renew the certificate of message bus.
- This script also supports renewal of certificates for any egress communication. If your IDP certificate is expired, you can replace or add a new certificate to the truststore of all services using this script.

Prerequisites

Here are the prerequisites:

- You should have a new SSL certificates that needs to be imported.
- All services must be running over SSL Terminate at ingress, except the message bus.

Renewing Ingress Certificates

To renew the ingress certificates:

1. Run the following to verify ingress certificates:

   ```
   $COMMON_CNTK/scripts/manage-certificates.sh -p $PROJECT -i $INSTANCE -c
   verify -t ingress
   ```

   This command shows the validity for all ingress certificates for all services. You can use `-a <servicename>` option in the above command to verify certificates for any particular service.

2. Run the following command to renew ingress certificates:

   ```
   $COMMON_CNTK/scripts/manage-certificates.sh -p $PROJECT -i $INSTANCE -c
   import -t ingress
   ```

   This command prompts for the certificate and key inputs. You should provide new certificates and then all ingress certificates will be renewed. You can also use `-a <servicename>` option to renew certificates for any particular service.

Import Egress Certificates

To import egress certificates:

Copyright © 2023, 2026, Oracle and/or its affiliates.

1. Run the following command to verify egress certificates:

   ```
   $COMMON_CNTK/scripts/manage-certificates.sh -p $PROJECT -i $INSTANCE -c
   verify -t egress
   ```

   This command shows the validity for all egress certificates from the truststore of all services.

2. Run the following command to renew egress certificates:

   ```
   $COMMON_CNTK/scripts/manage-certificates.sh -p $PROJECT -i $INSTANCE -c
   import -t egress
   ```

   This command prompts for the certificate and alias name inputs. You should provide the new certificate along with alias to store the certificate.

   > **Note**
   >
   > - If the provided alias name already exists, the older certificates will be overridden by the new certificate. Therefore, if you want to retain the old certificate, provide a new alias name.

Postrequisites

Following are the postrequisites:

- If you have imported egress certificates for any application, make sure you restart it.
- In case of message bus ingress certificate renewal, you must restart message bus to get changes reflected.
- After the renewal of ingress certificates, make sure that you have imported the new certificates into the client's trust.

Copyright © 2023, 2026, Oracle and/or its affiliates.