Event Driven Automation

Release 24.12

User Guide

3HE 20937 AAAB TQZZA
Edition: 1
December 2024

Nokia is committed to diversity and inclusion. We are continuously reviewing our customer documentation and consulting with standards bodies to ensure that terminology is inclusive and aligned with the industry. Our future customer documentation will be updated accordingly.

This document includes Nokia proprietary and confidential information, which may not be distributed or disclosed to any third parties without the prior written consent of Nokia.

This document is intended for use by Nokia’s customers (“You”/”Your”) in connection with a product purchased or licensed from any company within Nokia Group of Companies. Use this document as agreed. You agree to notify Nokia of any errors you may find in this document; however, should you elect to use this document for any purpose(s) for which it is not intended, You understand and warrant that any determinations You may make or actions You may take will be based upon Your independent judgment and analysis of the content of this document.

Nokia reserves the right to make changes to this document without notice. At all times, the controlling version is the one available on Nokia’s site.

No part of this document may be modified.

NO WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF AVAILABILITY, ACCURACY, RELIABILITY, TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, IS MADE IN RELATION TO THE CONTENT OF THIS DOCUMENT. IN NO EVENT WILL NOKIA BE LIABLE FOR ANY DAMAGES, INCLUDING BUT NOT LIMITED TO SPECIAL, DIRECT, INDIRECT, INCIDENTAL OR CONSEQUENTIAL OR ANY LOSSES, SUCH AS BUT NOT LIMITED TO LOSS OF PROFIT, REVENUE, BUSINESS INTERRUPTION, BUSINESS OPPORTUNITY OR DATA THAT MAY ARISE FROM THE USE OF THIS DOCUMENT OR THE INFORMATION IN IT, EVEN IN THE CASE OF ERRORS IN OR OMISSIONS FROM THIS DOCUMENT OR ITS CONTENT.

Table of contents

1 About this document

The Event Driven Automation (EDA) *User Guide* describes the system's core graphical user interface (GUI) and its use to configure and manage resources. It provides some information about EDA's purpose, a brief summary of its architectural components and central concepts, and includes administration concepts and procedures for maintaining EDA and the network elements it can manage.

Many of the capabilities of EDA are provided by individual applications that can be installed, updated, or uninstalled independently of the core EDA software. For details about those capabilities, see the documentation for individual applications.

This document is intended for network technicians, administrators, operators, service providers, and others who use EDA.

Note: This manual covers the current release and may also contain some content that will be released in later maintenance loads. See the *EDA Release Notes* for information about features supported in each load.

1.1 Precautionary and information messages

The following are information symbols used in the documentation.

DANGER: Danger warns that the described activity or situation may result in serious personal injury or death. An electric shock hazard could exist. Before you begin work on this equipment, be aware of hazards involving electrical circuitry, be familiar with networking environments, and implement accident prevention procedures.

WARNING: Warning indicates that the described activity or situation may, or will, cause equipment damage, serious performance problems, or loss of data.

Caution: Caution indicates that the described activity or situation may reduce your component or system performance.

Note: Note provides additional operational information.

Tip: Tip provides suggestions for use or best practices.

2 Introducing EDA

The Nokia Event Driven Automation framework (EDA) is an extensible, declarative, intent-based automation and operations NetOps platform that delivers agile and scalable network operations for data center and cloud environments of any size.

The EDA framework and capabilities include:

- A truly event-driven platform.
  Configuration is an event. A state transition in the network is an event. Operational activities are events. The controller generically handles these events, no matter their source, with the goal to reconcile the state of managed elements to get closer to the intended state.
- Declarative, extensible intent-based operations and automation across all aspects of device management.
  EDA simplifies and dramatically reduces effort across many operational and design tasks.
- An open framework to build event-driven applications.
  Applications built upon EDA's open framework can provide simple configuration normalization or templating, handle state updates, generate and process their own resources, raise alarms, publish and populate GUI views, and much, much more. All out-of-the-box configuration and state handling is built on this framework and is open-source.
- A stream-everything design.
  Receive on-change, real-time notifications on anything and everything in the system or network in the UI, over the API, or within custom event-driven applications.
- Built on cloud-native design principles.
  Scale flexibly with microservices based on the number of managed resources and endpoints, simplifying integrations and controller interactions by providing a single API and automation layer no matter your scale.
- Ubiquitous observability through “on-change” streaming telemetry from Nokia Service Router Linux (SR Linux).

  Every configuration and state path is available, all of the time.
- A query language and corresponding endpoint.
  The EDA query language (EQL) provides filtered, ordered, on-demand retrieval and streaming of any YANG path on any managed elements network-wide, or any system-published data.
- A Digital Sandbox.

  Enables a digital twin capability that emulates the network and is used to dramatically reduce both testing time and network risk.
- A Continuous Integration/Continuous Deployment (CI/CD) framework embodying DevOps principles applied to the network.

  Write custom pre and post-checks, test changes in branches, and follow the GitOps model for merging changes into production.
- Integration with major cloud management platforms.

  Integration enables the automation of network connectivity to support constantly changing application requirements.

- Resource abstraction and selection via labels.
  With EDA, you can flexibly label resources to dynamically update dependencies.

2.1 Fundamentals

Within EDA, controllers continually reconcile managed elements based on external or internal events in the system with the help of pluggable automation applications.

These applications define their own resources and define the logic these resources enact when created, modified, or deleted. This is similar to the Kubernetes controller loop, and event-driven applications work similarly to Custom Resource Definition-based controllers and their reconciliation loop in Kubernetes.

Where EDA differs from the reconciliation loop is when only a partial state can be achieved. EDA uses a transaction concept for rolling changes onto groups of network elements. Transactions are flexible in their boundaries, allowing you to group changes that must succeed together into a single transaction, and avoid having to deal with ordering complexities or partially deployed changes.

Apps built on EDA leverage a generic intent framework that allows the building of event-driven applications to implement intents. The architecture allows custom resources with a user-defined schema to be loaded, and to have customized logic written in MicroPython that handles the creation and updates of these resources. Where this differs from other controllers is that applications are entirely stateless, run-to-completion entities, with the EDA ConfigEngine (or CE) automatically handling deltas between executions. Apps are simple to write, and can be iterated on using a CI/CD workflow in production.

Going beyond simple customizations, EDA provides a full-featured platform for application development. Application writers can build their own UIs using the EDA Rapid Application Development (ERAD) framework and handle state transitions of any YANG path in the MicroPython framework. Applications are able to generate alarms, normalize data and write it back to the EDB, or simply update the status of a resource, and more.

EDA takes CI/CD and the infra-as-code movement to heart. On the completion of a transaction, the resulting set of input resources are written to a local git repository and can be pushed to one or more remotes, with this being the only required persistent database. This unique approach allows revision control on all resources in the system, a simplified backup and restore approach, and a simplified geo-redundancy model, with two or more sites configured with the same repository.

Additionally, using git as a database allows the true CI/CD workflow in the modification of network resources. EDA supports a branching concept, in which one or more additional EDA instances can be instantiated on a branch off a main, production cluster. Changes can then be made on the branched cluster before merging them back using merge/pull requests into the main branch.

Merging into main in this context is equivalent to pushing changes into production. But before this, you can use the Digital Sandbox to test the impact of any changes in a digital twin of the production network, including simulated nodes.

Importance of an open, model-driven, consumable NOS

The declarative, intent-based framework and the automation capabilities of the Nokia Event Driven Automation framework are only made possible by leveraging a modern NOS that offers an open, model-driven, stream-anything foundation.

Nokia SR Linux couples each service with its own YANG data model, allowing for broad, deep and efficient network data access from any interface. This modular approach provides open streaming telemetry and network management that use gNMI (gRPC Network Management Interface) to stream network data and configure network devices.

By using this modern approach for the NOS, the Event Driven Automation framework has timely and efficient access to more granular data across the entire fabric. This data can then be used to understand the state of the network, which is essential for event-driven applications to determine if the network is behaving according to their intent. This approach is also highly scalable, which is essential in today’s networks.

2.2 EDA as part of the Nokia Data Center Fabric solution

Since the advent of software-defined networking (SDN), the industry has experimented with the separation of the management, control, and data planes. For scalable and faster convergence in large-scale data center networks, the architectural approach that has gained industry momentum is a combination of distributed routing (using control planes and data planes) running on data center switches with the necessary network-wide control and automation functions implemented in an external controller. This approach combines the centralized control and programmability of a traditional SDN approach with the higher scalability and convergence of a distributed routing approach. This is the approach we have taken with the Nokia Data Center Fabric solution.

The Event Driven Automation framework is a key component of the Data Center Fabric solution, which also includes the following products:

- Nokia Service Router Linux (SR Linux): An open, extensible, and model-driven network operating system (NOS) based on Linux® that enables scalability, flexibility, and efficiency in data center and cloud environments.
- Nokia Data Center Fabric hardware platforms: A portfolio of next-generation leaf and spine switches that deliver massive scalability and performance while aggregating and interconnecting data center and cloud environments.

In support of fabric management, the Fabrics application exists to support the automation and operation of fabric topologies. Beyond this, EDA uses other generic applications to abstract overlay services, interfaces, underlay configurations, and so on.

2.3 Architecture components

The Nokia Event Driven Automation framework adopts a microservices design that is built on top of Kubernetes. These include both components native to EDA, and industry-standard third-party components.

2.3.1 EDA components

APIServer

APIServer is the gateway into EDA. APIServer dynamically extends its endpoint coverage as new event-driven applications are onboarded, and provides streaming endpoints for all resources. APIServer also acts as the common authorization point for RBAC, providing a common AAA mechanism across all endpoints. AAA itself is implemented using Keycloak.

App store

EDA uses a variety of supporting applications to manage its wide range of capabilities. EDA uses an on-product App store to display the set of available EDA applications, and assist in their installation, update, and removal. The App store also tracks the status of all installed applications.

ArtifactServer

ArtifactServer is the artifact storage server. ArtifactServer provides a flexible artifact cache in the cluster, allowing network elements and API clients to store and retrieve artifacts of many kinds.

Bootstrap server - Zero-touch Provisioning (ZTP)

ZTP performs bootstrap and discovery of managed endpoints, allowing an end user to implement a simple “plug-and-power-up” approach to onboard new devices onto the controller and surrounding network.

ConfigEngine

ConfigEngine is the core of configuration in EDA. It is responsible for reconciling all configuration input to the system, and executing any EDA App logic before pushing changes to NPP instances as needed. ConfigEngine uses a unique dependency model that allows applications to simply request information they need to reconcile. These requests build a dependency tree that can then be triggered if any dependent resources update. All updates to resources trigger their dependent resources (and their dependent resources in turn) to execute. This dependency logic goes as far as allocation pools, meaning that simply changing an IP pool in use from IPv4 to IPv6 would result in all users of the pool being triggered to process the update. You can convert your entire network from IPv4 to IPv6 in one reversible transaction.

Digital Sandbox

One of the key requirements for modern data centers is the ability to make faster, periodic changes while still managing the risk of a change. To this end, the EDA framework delivers a cloud-native Digital Sandbox, a containerized virtual infrastructure that emulates the production network by creating its digital twin. EDA uses the Digital Sandbox to validate intent across the life cycle of the fabric. The Digital Sandbox is also an essential part of the network validation phase of the CI/CD pipeline process. The EDA framework integrates the Digital Sandbox in all its workflows to provide design validation and change management flexibility, thereby reducing the risk of changes in a dynamic data center environment. With this capability, operators can make faster, periodic, and independent changes to the network, lowering risk and increasing operational agility.

The Digital Sandbox is an integral part of Nokia’s approach for CI/CD and is used to trial and validate network changes before deploying them in the production network. Changes can include initial fabric design, initial service connectivity, software upgrades, introduction of new devices, policy configuration changes, and failure scenarios.

The Digital Sandbox provides a digital twin of the data center fabric, emulating network elements by deploying a containerized Nokia SR Linux and SR OS instance of each, which are used to test and validate any planned network changes.

The Digital Sandbox leverages on-change telemetry to maintain absolute parity of the network in configuration, routing, and state. It also can emulate external Border Gateway Protocol (BGP) speakers and generate synthetic traffic.

The Digital Sandbox allows any changes to the production network to be tested and validated before being deployed in the network, greatly reducing risk.

Some of the benefits of Digital Sandbox are:

- Time and resource savings: Saves time and resources by quickly and efficiently testing network, configuration and routing scenarios in a virtualized, pre-built environment that is in absolute parity with the network.
- Reduced risk: Greatly reduces risk to the network by first validating network changes in a fully emulated environment before deploying the changes in the network.
- Lower lab expenses: Reduces the effort and cost of setting up lab environments to test and validate network changes.
- Reduced power consumption: Drives a green approach to testing and validation by leveraging a virtualized environment that can be set up and changed in minutes.
- Ease of use: A complete virtual infrastructure is built into the Event Driven Automation framework and is fully programmable and easily set up through an intuitive UI.

State Aggregator

EDA's StateAggregator acts as an aggregation point for EQL queries and state requests. It provides a single, uniform interface to the EDA database (EDB).

StateAggregator supports multiple instances using replicas; this is controlled using `EngineConfig`. It supports demuxing requests into request to multiple shards, and aggregating results.

StateAggregator provides gRPC endpoints, secured using mTLS, to support the following functions:

- Verifying the liveness of SA.
- Executing and auto-completing EQL queries.
- Retrieving data from one or more shards matching filters, with various types of streaming and one-shot.
- Retrieving and streaming the schema of any table.

StateController

StateController is the core of state queries in EDA. StateController provides a scalable common layer for EDB queries, and maintains the shard map - or the locations of other EDB shards.

StateEngine

StateEngine is the core of state processing in EDA, responsible for reconciling all state input to the system, and executing any EDA app logic before pushing deltas to EDB to trigger any dependent applications.

Applications

EDA apps are a set of applications that provide several types of intent, including Fabric (Day 0 design), VirtualNetwork (Day 1 deployment), and maintenance intents (Day 2+ operations). The intent framework of the Event Driven Automation framework allows operators to define, in an abstract manner, the intended end state of resources and configuration. By using streaming telemetry to understand the current state, the system can determine any discrepancies from the intended state and implement any required network changes.

UI

EDA employs a simple, extensible, easy-to-use UI that allows for complete programmable operation and visualizations. Operations that can be performed through the UI can also be performed through EDA's REST APIs.

NPP

NodePushPull is a purpose-built microservice that ingests the highly scalable streaming telemetry offered by SR Linux and SR OS, and manages configuration interactions with network elements.

EDB

EDA DB is a purpose-built, sharded database that allows the distributed streaming and processing of state information in the cluster.

ERAD

EDA Rapid Application Development environment allows you to build any UI view you like with drag-and-drop components and flexible streaming queries to EDB.

Connect

Connect performs integration with cloud management platforms, allowing virtual machine (VM) or container “spin up” and “tear down” events to drive network change. This capability enables the data center fabric to react to workload and compute connectivity requirements. Connect integrates using REST APIs and a plugin-based model, enabling seamless, modular, and simple integration with cloud management platforms.

Workflow Engine

In EDA, both workflow and CI/CD functionality is supported through the WorkflowEngine. The WorkflowEngine acts as the controller behind the instantiation, status, and interaction with the `Workflow` and `WorkflowDefinition` resources.

2.3.2 Third-party components

In addition to its own internal, native components, EDA uses a collection of well-known, industry-standard third-party components to support its operations.

Kubernetes

Kubernetes provides an event-driven microservices foundation. Running natively in Kubernetes has numerous benefits, including providing abstraction from physical compute resources, as well as the ability to define the entire deployment through infrastructure-as-code (IaC) principles using Helm charts and kpt as package managers.

Cert Manager

EDA uses CertManager, an extensible X.509 certificate controller, to generate, sign, and distribute the signed certificates and keys for pods. CertManager validates certificates for public and private issuers, and can assist in renewing certificates before they expire.

Fluentd and FluentBit

EDA uses Fluentd and FluentBit to assist with the processing of logging data. EDA uses FluentBit to collect logs from worker nodes, and uses Fluentd as an aggregator. By default, the embedded instance of Fluentd logs API hits in their own file, logs application data in their own files, and maintains an aggregate errors log.

Git

EDA uses Git to store data for all successful transactions. This allows EDA to support reverting of transactions and restoring to previous states. Most often Git capabilities are used to support revision control over individual resources or sets of resources.

Keycloak

EDA uses KeyCloak to support authentication, and passes authentication to KeyCloak directly instead of using intermediate APIs. Authentication (and subsequent authorization) are required to interact with all non-login APIs in EDA.

Metrics Server

EDA uses Kubernetes Metrics Server to collect and share resource metrics.

2.4 Declarative, intent-based automation

The Nokia Event Driven Automation framework allows operators to represent the configuration and initial state of the data center fabric in a declarative, intent-based way. With this declarative approach, the desired configuration and state of the fabric can be specified up front in a simplified or abstract way that defines how the fabric should operate. This intended state, which is stored centrally, represents “the single source of truth” and can be used to iteratively validate the actual state of the network.

Day 0 intent-driven design

By taking an abstract, intent-based approach for Day 0 design, the data center operator can focus on high-level aspects of the design, identifying the minimal information needed to build a data center fabric. For example, the operator needs to input only a few parameters, such as the number of racks and the number of servers per rack.

The system uses this information to automatically generate the rest of the detailed configuration based on Nokia-certified design templates. The result is a standard BGP-based IP fabric design (for example, number of racks, number of servers per rack, IPv4/IPv6 addressing, BGP configuration, cable map, and so on) that can be validated using the Digital Sandbox before being deployed to the data center fabric.

With this intent-based approach, multiple leaf-spine fabrics can be created by easily replicating the first one created or by creating a customized fabric.

Day 1 intent-driven deployment

For Day 1 deployment, one of the initial tasks performed by the Event Driven Automation framework is fabric discovery and node bootstrap. The Event Driven Automation framework offers Zero Touch Provisioning (ZTP) to turn up new leafs and spines, allowing the adoption of a simple plug-and-power-up approach to onboard new nodes onto the fabric.

After the new nodes are onboarded, the Event Driven Automation framework can then push Day 0’s validated design to the fabric, thereby completing deployment of the initial network underlay portion of the fabric.

Day 1 deployment uses the concept of Virtual Networks to automate the creation of the required overlay connectivity, to support the initial application workloads that are hosted on attached compute resources. To create this connectivity, the Event Driven Automation framework leverages Ethernet VPN (EVPN) Layer-2 and Layer-3 services within and across the data center fabric.

The Virtual Network application abstracts the complexity of the EVPN configuration by enabling the data center operator to focus on specifying high-level parameters. This high-level intent can be as simple as identifying the set of downlinks an application workload uses to connect to the fabric. Virtual networks can be validated using the Digital Sandbox before being deployed into the production network.

Complexities such as switch-to-switch EVPN and allocation of VXLAN network identifiers, route distinguishers, route targets, Ethernet segment IDs and Ethernet virtual interfaces are all abstracted and left to the Event Driven Automation framework to generate according to the high-level intent parameters specified by the operator.

Day 2+ intent-driven maintenance

During Day 2+ operations, the EDA framework uses maintenance intents (such as hardware intents and software intents) to define the intended state of the network in terms of software and hardware. With these two intent types defined, the desired software load and hardware version across the network is defined for each leaf, spine, or super-spine switch.

For Day 2+ operations, the EDA framework constantly monitors the fabric by leveraging on-change telemetry it receives directly from various sources in the network. The EDA framework compares this information with various intents and analyzes the results to find configuration inconsistencies, faults or other deviations that may lead to network issues.

Each inconsistency, fault, or other deviation is flagged and presented to the operator to be either accepted or rejected. These inconsistencies can often require a change to the network (for example, a configuration change or software upgrade) to fix the problem. With the EDA framework, the operator can automate the testing and validation of these network changes using the Digital Sandbox. If these changes pass validation, they can be scheduled for automatic deployment into the production network.

This process of automated testing and validation dramatically lowers the risk of deploying network changes because it identifies any potential problems before a change is deployed in the network.

2.5 Fabric observability

To operate today’s modern data center fabrics, real-time observability information is required to support various operational tasks. Fabric observability is needed to monitor the fabric, and is achieved by accessing a combination of on-change, streaming telemetry and log data that represents the network state and is collected directly from the data center fabric.

Multi-dimensional telemetry comes from various sources, including:

- Basic telemetry: Faults, standard statistics, TCAM/LPM, and so on.
- The control plane: Link Layer Discovery Protocol/Link Aggregation Control Protocol (LLDP/LACP) state and events, BGP adjacency, BGP routing information base (RIB), forwarding information base (FIB), and so on.
- The fabric workload layer: Topology, number of apps, number of flows, and so on.

The EDA framework constantly receives this information using the SR Linux gNMI and leverages instances of NPP to ingest this streaming telemetry while scaling as required. The EDA framework enables a cloud-native, scale-out collector architecture to ensure that collection capabilities are highly distributed.

2.6 Fabric operations

After the data center fabric is designed and deployed, the Day 2+ operations phase begins. In this phase the Event Driven Automation framework compares both design and workload intent (that is, single source of truth) with all the telemetry data collected from the fabric to both optimize operational tasks and ensure that the network is operating as expected.

Fabric integrations

The EDA framework provides an open REST API that allows third parties to have full access to the system. A flexible, cloud-native approach enables integration of the EDA framework with many different customer cloud environments.

Cloud-native architectures, built with microservices and containers, are pushing the limits of network scalability and performance, requiring networks to be much more responsive to changes in applications. Modern data center fabrics need to be synchronized with applications to remove the network as an obstacle to innovation. There needs to be a symbiotic relationship between the applications and the network that serves them.

To tackle this requirement head-on, the EDA framework has implemented a Connect microservice that allows for integration, using a plugin infrastructure, with cloud management platforms such as OpenStack, VMware vSphere, and Kubernetes. With this integration, any change events to workloads (both virtualized network functions and containerized network functions) are immediately understood by the Connect service. This allows the fabric to react in real-time to these events and ensures that Layer-2 and Layer-3 fabric connectivity always supports these changes. This type of integration is essential to scale next-generation data center networks.

2.7 Conclusion

As the demands on data center networks continue to drive openness, flexibility, and agility, the Nokia Data Center Fabric solution was purpose-built to meet these challenges. As part of this solution, the Nokia Event Driven Automation framework delivers declarative, abstract intent where automation and simplification are needed while also delivering detailed insights by monitoring every aspect of the data center fabric. This combination of abstract intent-based automation plus detailed openness and visibility allows the data center operator to perform Day 0 design, Day 1 deployment, and Day 2+ configuration, operation, measurement, and analysis of a data center fabric.

3 GUI basics

This section describes the basics of using the EDA GUI: signing in, signing out, and working with common GUI elements.

3.1 Signing in

Prerequisites

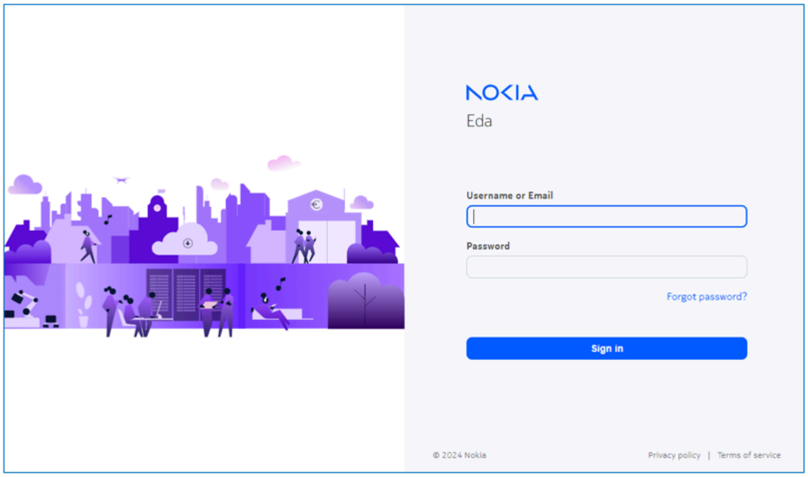
To log into the EDA GUI, you must have:

- the URL to the EDA server; for example, https://<server URL>:9443/
- a valid user account and password for EDA

Procedure

Step 1. Go to the URL for the EDA server, in the form of: https://<server URL>:9443/

Figure 1: The EDA login page



Step 2. Enter your username or email address.

Step 3. Enter your password.

Note: If you have forgotten your password, click the **Forgot password?** link. Enter your email address in the field displayed and click **Submit** to receive instructions for resetting your password.

Step 4. Click **Sign in**.

Note: If more than 15 minutes has elapsed from when you opened the login page and when you click Submit, EDA rejects your login attempt and display an error message indicating that you took too long to log in. If this happens, enter your credentials again and click **Submit** to log in.

Expected outcome

You are logged into the EDA GUI. If configured by your administrator, a login banner displays with information for all EDA users.

Note: You can open multiple tabs displaying the EDA UI in your browser. If at any time any of the EDA session tabs has been idle for 15 minutes, a message displays in the active tab indicating this. It offers the options to either sign out, or remain signed in. If you do not choose to remain signed in within 30 seconds, EDA automatically signs you out.

3.2 Signing out

Procedure

Step 1. Click the **User** icon at the upper right of the EDA GUI.

Step 2. Click **Sign Out**.

Step 3. At the confirmation prompt, click **Sign Out** to finish logging out, or **Cancel** to remain logged into EDA.

Note: If you delay your choice for more than one minute, EDA signs you out automatically.

3.3 Home page

The EDA Home page is your first landing page for the EDA GUI. While some elements of the Home page are unique, it also includes a set of standard controls that are available from all pages within the EDA GUI.

Figure 2: The EDA Home page

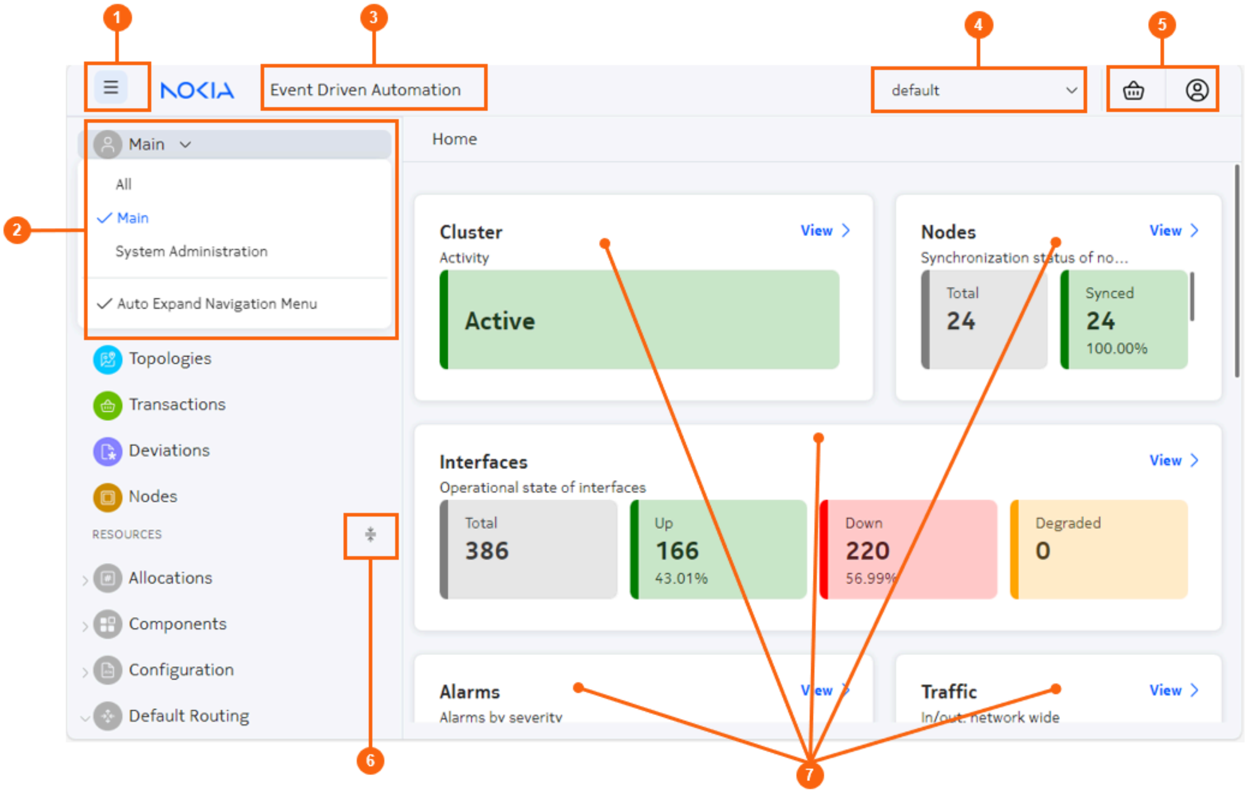


Table 1: Common elements on the Home page

#	Name	Function
1	Menu pin/un-pin	Use this button to expand and pin, or collapse and hide, the main menu in the left column of the EDA GUI.
2	Menu selector	Use this drop-down selector to toggle between: • the **Main** EDA menu, providing access to network management functions • the **System Administration** menu, providing access to functions that manage EDA itself • **All**, which displays all menu selections for the Main and System Administration menus .
3	Event Driven Automation Home button	Click here to navigate to, or re-load, the EDA Home page.
4	Namespace selector	Use this drop-down selector to choose a working namespace: either all namespaces, or one specific namespace. This selection affects

#	Name	Function
		the namespace from which to display data, and either create or manage resources.
5	Common buttons	• The **Transaction** basket: indicates the number of pending transactions for the current user. Click to open the **Transactions** panel. • **User settings** button: click to open the **User Settings** menu.
6	Menu heading expand/ collapse toggle	Click this control to alternately expand all headings or collapse all headings in the main navigation menu.
7	Panels	Each panel displays important information about the status of the EDA application and the network it is managing. Clicking the View link in any panel opens the EDA GUI page specific to that panel's information.

Related topics

Default Home page panels

Transactions

User settings

Information panel

3.3.1 Default Home page panels

Note: All panels are "live"; they continuously update to show the latest data.

- **Cluster**: the current cluster and status.
- **Nodes**: displays the synchronization state of the nodes known to EDA (total nodes; synced nodes; unsynced nodes).
- **Interfaces**: displays the operational state of the interfaces known to EDA (total interfaces, Up interfaces; Down interfaces).
- **Alarms**: displays the number of current alarms, and their percentage distribution by alarm type.
- **Traffic**: displays total inbound and outbound traffic for the network as a whole.
- **Top interfaces - In**: displays the five interfaces exhibiting the most incoming traffic.
- **Top interfaces - Out**: displays the five interfaces exhibiting the most outgoing traffic.

3.4 GUI elements

The following elements are available at the upper right or right edge of every page within the EDA GUI.

3.4.1 Transactions

A a transaction defines a set of changes that need to occur synchronously in EDA.

Click the Transactions basket icon to open the Transactions drop-down panel.

Figure 3: Transactions button

The Transactions drop-down panel displays information and options for the current transaction. From here you can edit the configuration changes contained within the transaction, discard the transaction, or commit the transaction.

Related topics

Transactions

3.4.2 User settings

Opening the User drop-down panel displays information and available actions for the currently logged-in user.

Figure 4: User settings button

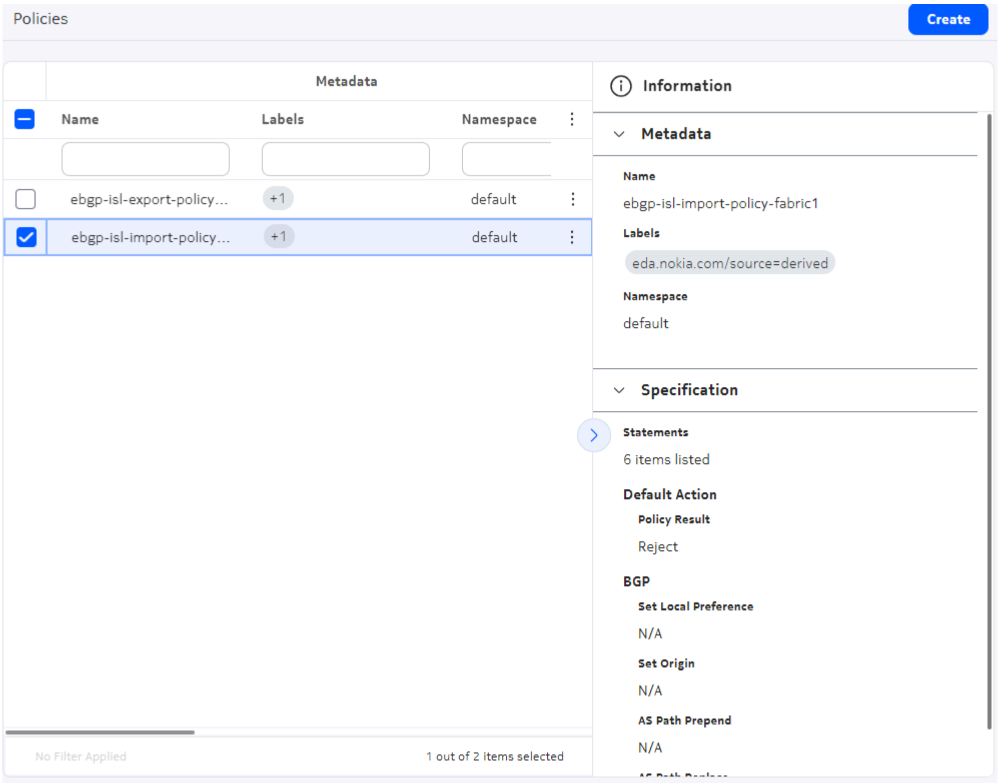
- **User information**: displays the name of the currently logged-in user, that user's Role, and the date of the last successful login.
- **Appearance Theme**: click to select a display theme from among the following:
  - Light: primarily displays dark text on a light background
  - Dark: primarily displays light text on a dark background
  - Enhanced Dark: like Dark, but employs even darker background shading
- **Change Password**: click to change the password for the current user. You must re-authenticate before you can complete the password change.
- **Release Information**: click to view the release number for this version of the EDA software.
- **Sign Out**: click to sign out of the EDA application.

3.4.3 Information panel

Most pages in the EDA GUI include an **Information** panel. You can open this panel by clicking the **Expand/Contract** control at the middle right of any page.

The Information panel displays information about any selected object on the corresponding main page.

Figure 5: An example of an expanded information panel, showing the collapse control



Policies

	Name	Labels	Namespace
□	ebgp-isl-export-policy...	+1	default
☑	ebgp-isl-import-policy...	+1	default

No Filter Applied

1 out of 2 items selected

Information

Metadata

Name
ebgp-isl-import-policy-fabric1

Labels
eda.nokia.com/source=derived

Namespace
default

Specification

Statements
6 items listed

Default Action
Policy Result
Reject

BGP
Set Local Preference
N/A
Set Origin
N/A
AS Path Prepend
N/A

3.5 Key bindings

In the EDA UI, you can perform some common actions by pressing a fixed set of keys or key combinations.

Where a combination is shown as <key> + <key>, press the second key within two seconds of the first key to perform the action.

Where a combination is shown as <key> - <key>, press both keys at the same time to perform the action.

Table 2: Global shortcuts (available from all pages)

Key combination	Action
g+h	Navigates to home screen
g+a	Navigates to the alarms screen
g+q	Navigates to the queries screen
g+t	Navigates to the topologies screen
g+r	Navigates to the transactions

Table 3: Resource list shortcuts

Key	Action
e	While an instance of a resource is selected, pressing e allows you to edit the form.
d	Deletes (with warning) the instance of the selected resource.
c	Brings up the form to create a new instance of the resource.
i	Brings out the information panel if closed; closes the information panel if open.
a	Auto-resizes the columns.
m	Brings up the **manage column** form.
[	Selects the first item in the list if none are selected. If an item is already selected, moves the focus to the previous row if one is available.
]	Selects the first time in the list if none are selected. If an item is already selected, moves the focus down to the next row if one is available.
space	Toggles selection and de-selection of the active or selected row.

Table 4: Form shortcuts

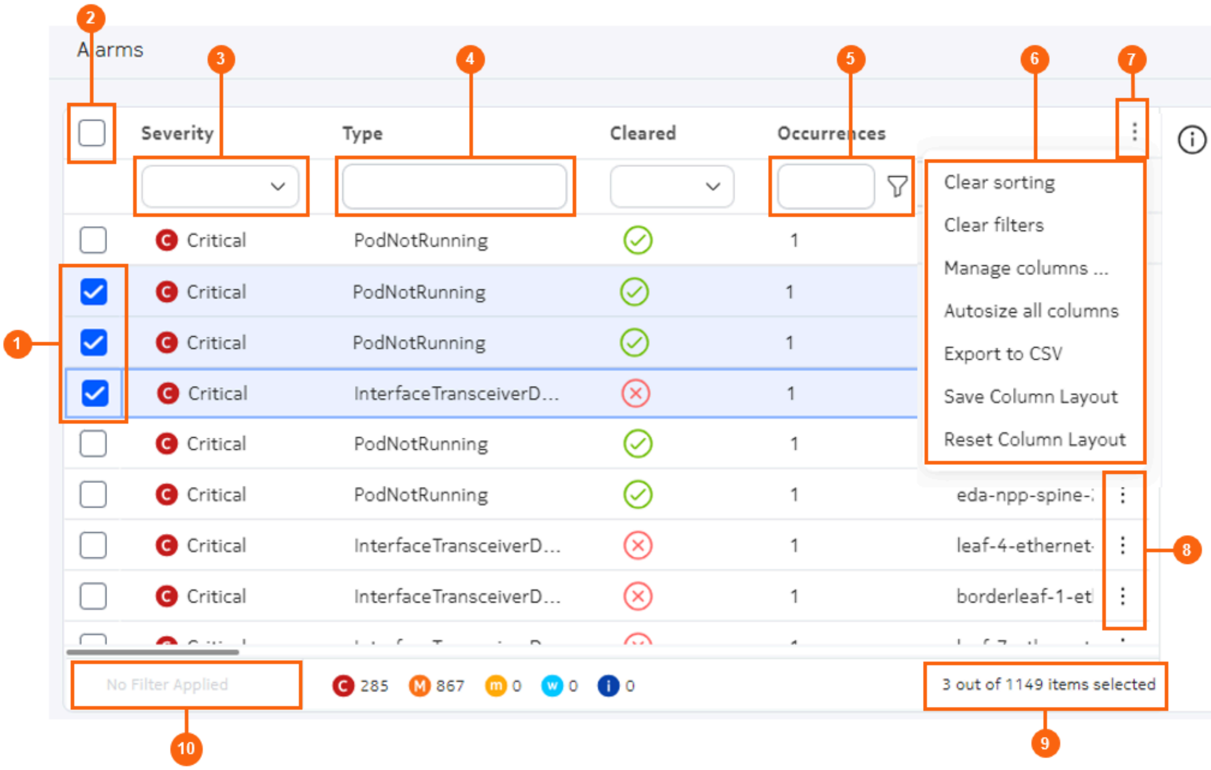
Key	Action
shift-y	Toggle the YAML view open or close
shift-d	Toggle the descriptions of the properties
shift-r	Toggle required fields on/off
shift-f	Takes cursor to the field search box
shift-t	Add form to transaction
shift-c	Commit the form

3.6 Working with data grids

Many pages in the EDA GUI display lists of featuring rows and columns of data. The options described here for managing such data grids are common to most data grids in the EDA GUI.

Note: Many lists include special options uniquely available for lists of particular data. Those options are described in the topics pertaining to those lists.

Figure 6: A sample list with controls



Alarms

☐	Severity	Type	Cleared	Occurrences	
☐	Critical	PodNotRunning	✓	1	
☑	Critical	PodNotRunning	✓	1	
☑	Critical	PodNotRunning	✓	1	
☑	Critical	InterfaceTransceiverD...	✗	1	
☐	Critical	PodNotRunning	✓	1	
☐	Critical	PodNotRunning	✓	1	eda-npp-spine-:
☐	Critical	InterfaceTransceiverD...	✗	1	leaf-4-ethernet
☐	Critical	InterfaceTransceiverD...	✗	1	borderleaf-1-et

No Filter Applied C 285 M 867 m 0 w 0 i 0 3 out of 1149 items selected

Table 5: List controls

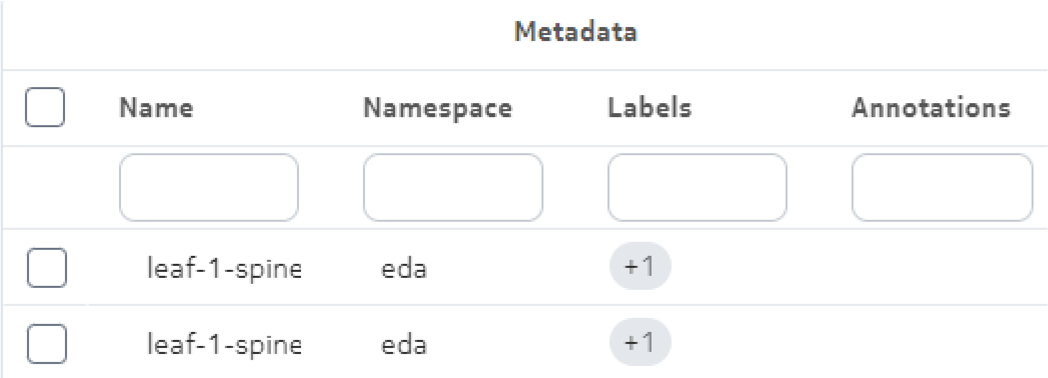
#	Name	Function
1	Row selectors	Use these to select one or more individual rows.
2	Multiple row selector	Use this to toggle between selecting all rows, and un-selecting all rows.
3	Selection filter	Select a value to filter the list based on matching values in that column.
4	Text string filter	Enter an alphanumeric string to filter the list based on matching values in that column.
5	Logical filter (with filter menu icon)	Create a logical expression, which can include multiple criteria and a choice between AND and OR requirements, to filter the list based on matching values in that column.
6	Standard table actions	Select from a set of actions common to all tables.
7	Table settings and actions button	Click to open the menu of standard table actions.
8	Table row actions button	Click to select from a set of actions specific to the data in that row.

#	Name	Function
9	Row and selection counter	Indicates the total number of rows, and the number of rows that are currently selected.
10	Filter counter	Indicates how many filters have been applied to result in the currently displayed data set.

Nested columns

Columns may be collected into groups; in this case, individual columns are nested within a higher-level column for that group. For example, several columns pertaining to resource metadata are nested within a **Metadata** column.

Figure 7: Nested columns for metadata



	Metadata			
□	Name	Namespace	Labels	Annotations
□	leaf-1-spine	eda	+1	
□	leaf-1-spine	eda	+1	

Managing displayed columns

For any table, you can select which columns are displayed and which are hidden from view.

In the list of standard table actions, click **Manage columns...** to open a Manage Columns dialog. This dialog lists all available columns; those checked are included in the data grid, and those unchecked are excluded. By default, some possible columns may already be excluded from view.

Select or un-select the available columns and click **Apply** to close the dialog and update the data grid display based on your selections.

To restore the default configuration for the data grid, click the **Table settings & actions** icon and select **Reset Column Layout** from the action list.

If the set of columns exceeds what can be shown at one time in the display area, the EDA UI adds a scroll bar to the bottom of the data grid. Scrolling horizontally moves all columns to the left or right as you would expect.

By default, the **Name** and **Namespace** columns are pinned to the left for all data grids that include them. When scrolling horizontally, these columns remain visible at the left edge of the display.

Sorting

For any table, you can sort the row order based on the values in any column by clicking on the title for that column. EDA displays a sorting icon next to the column title to indicate that sorting is active.

Clicking on the title again toggles between ascending and descending order.

To sort by multiple columns, shift-click a series of column titles. Doing so has the following effects:

- adds each successive column to the sort order
- displays the sorting icon next to each column title
- displays a number next to each column title to indicate its rank in the overall sorting order

To clear all sorting from the data grid, click the **Table settings & actions** icon and select **Clear sorting** from the action list.

Filtering

You can filter the displayed data to include only those with specific values in one or more columns:

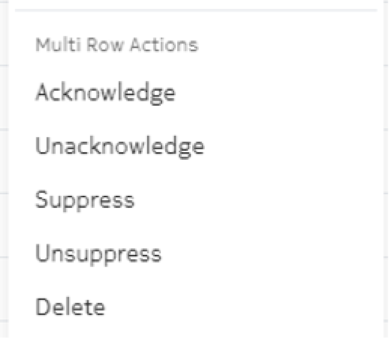
- For columns that display text, you can type any alphanumeric string in the field at the top of the column. The list is filtered only to show rows with the selected value in that column.
- For columns that display only a predetermined set of values, you can use a drop-down list to select a value. The list is filtered only to show rows with the selected value in that column.
- For columns that display numbers, you can click the **Filter** icon to build a logical filter. The filter menu allows you to choose an operator and a value, and then add additional operator/value combinations to create a complete logical expression. The list is filtered only to show rows with the selected value in that column.

To clear all filtering from the data grid, click the **Table settings & actions** icon and select **Clear filters** from the action list.

Multi-row actions

Some tables support actions that can be simultaneously applied to all selected rows. When available, these actions are displayed under a submenu of the **Table settings and actions** menu.

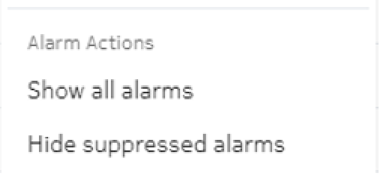
Figure 8: Multi-row actions for the alarms list



Special actions

Some tables support special actions appropriate to the particular data displayed in the list. When available, these actions are displayed under a submenu of the **Table settings and actions** menu.

Figure 9: Special actions for the alarms list



Related topics

Key bindings

Namespaces

3.7 Common list actions menu

The following options are available for the menu of **Table settings and actions** for any table in the EDA GUI.

- **Clear sorting**: select to remove all sorting from all columns.
- **Clear filters**: select to remove all filters from all columns.
- **Manage columns...**: for any table, the displayed columns may be a subset of all available columns. Select this option to view a list of all possible columns, and enable or disable any items.
- **Autosize all columns**: if you have previously adjusted the width of any column, select this option to restore all columns to their default width.
- **Export to CSV**: select to save a comma-separated-values (.csv) file containing all data for the currently displayed table. The file is saved to your default download directory.
- **Save Column Layout**: select to save the column selection and column width for the table on this page. These settings are saved as part of your user account and are retained in future sessions until you change them or reset the column layout.
- **Reset Column Layout**: select to restore the column layout to the default settings.

4 Dashboard designer

In EDA, you can construct your own dashboard pages to display the data you deem important for your operation.

The Dashboards page allows you to construct a dashboard, which itself can contain one or more layouts. Each layout is a separate dashboard page, selectable using a drop-down on the main dashboard.

Each dashboard layout can be either a page consisting of a collection of dashlets, each displaying its own source data; or a single list, which is a conventional data grid displaying a single set of source data.

For dashboard layouts, you can construct each layout by selecting from a set of pre-defined dashlets. Each dashlet can show a particular type of data like counters, lists, and charts. You can then add these dashlets to a page you design, optionally distributing them among a set of rows and columns you have specified within the page. Individual dashlets can be set to span multiple rows or columns.

For each list layout or dashlet layout in the dashboard, you define source data by constructing an EDA query, natural language query, GVK definition, or specifying a URL endpoint. You can then specify details about which parts of that data are displayed, set thresholds for highlighting, and make other formatting choices depending on the dashlet type.

The Dashboard designer only displays dashboards that were created by the current user. A future release will allow users to publish and share dashboards with other users.

4.1 Dashboards page

The Dashboards page is the point from which you can access existing dashboards, and begin creating new dashboards.

Figure 10: The dashboards page

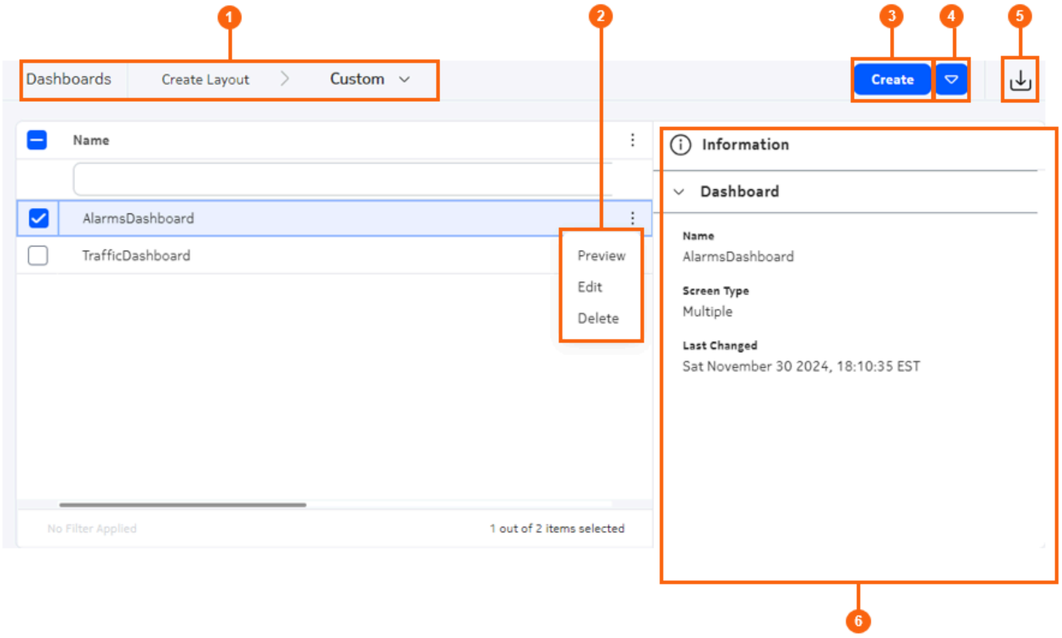


Table 6: Elements of the Dashboards page

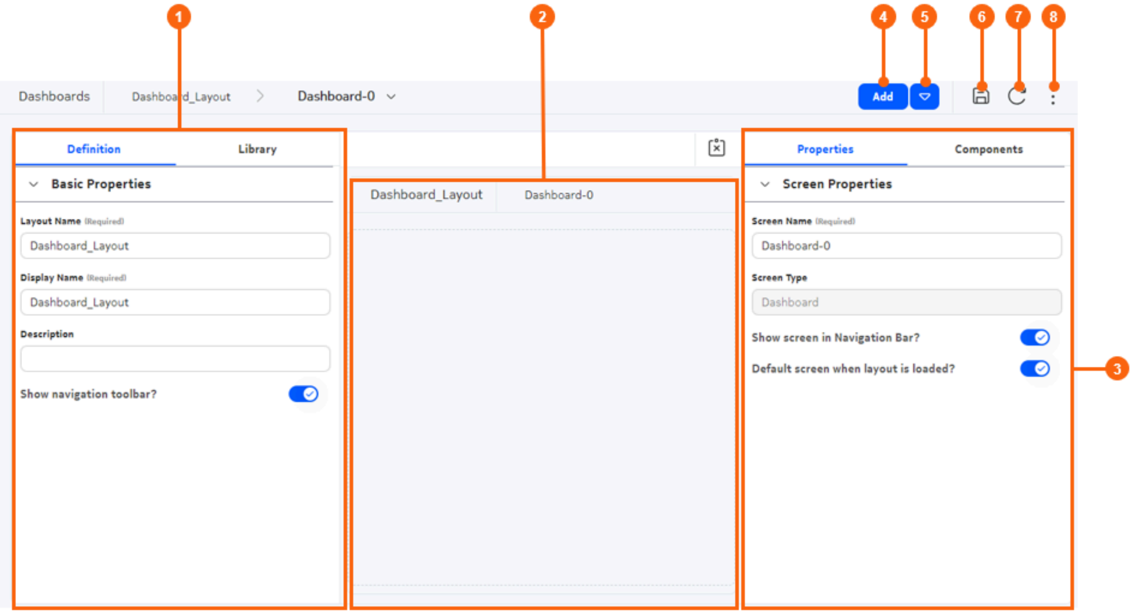
#	Name	Function
1	Breadcrumb bar	Displays the current position in the set of Dashboard designer pages. Clicking on the Custom part of the Breadcrumb bar displays a drop-down that includes a Core option. The Core designer is not yet enabled, and selecting **Core** in this drop-down results in a message to that effect.
2	Row actions	Clicking the Table row actions icon reveals the actions available for dashboards displays in the data grid. • Preview: Displays a preview of the selected dashboard. • Edit: Opens the Dashboard designer view for this dashboard. • Delete: Deletes the selected dashboard.
3	Create	Click to open the Dashboard Designer view for a new dashboard.
4	Create List Layout	Click to open the Dashboard Designer view for a new dashboard consisting of a single list.
5	Import	Click to open a file selection dialog. Select the file for a previously exported dashboard and click **Open** to import the selected dashboard. The new dashboard is then displayed in the Dashboards list.

#	Name	Function
6	Information panel	A standard EDA information panel, displaying details about the selected dashboard.

4.2 Dashboard designer page

The dashboard designer page is the space in which to create a dashboard and its constituent layouts, and to configure the data displayed on each.

Figure 11: The dashboard designer page



Basic Properties

Layout Name (Required): Dashboard_Layout

Display Name (Required): Dashboard_Layout

Description:

Show navigation toolbar? ✓

Dashboard_Layout Dashboard-0

Screen Properties

Screen Name (Required): Dashboard-0

Screen Type: Dashboard

Show screen in Navigation Bar? ✓

Default screen when layout is loaded? ✓

Table 7: Elements of the Dashboard Designer page

#	Name	Function
1	Definition/Libraries panel	The Definition tab displays basic parameters about the current dashboard layout, like its name and description. The Library tab displays elements that you can add to the current dashboard layout: flex rows, flex columns, and dashlets.
2	Layout panel	This is the area that displays the dashboard layout you are designing. Drag objects from the Libraries tab into this space to add elements to the dashboard layout. Select objects in this panel to view and configure their properties in the Properties tab.

#	Name	Function
3	Properties/Components panel	The Properties tab displays properties for the current dashboard, and for the row, column, or dashlet currently selected in the layout panel. Use this tab to configure the dashboard, and to configure the behavior of dashlets, including the query or other data source for the dashlet.
4	Add	Click to add another dashboard layout to the dashboard.
5	Add List Layout	Select this option in the drop-down to add a List page to the dashboard.
6	Save	Click to save the current dashboard design.
7	Reset	Click to discard all changes since you last saved the layout, after confirmation.
8	More icon	Click to view a list of available actions for the current Dashboard: • **Preview saved changes**: open a new tab that displays the current dashboard design. • **Export**: save the dashboard design as a file, which others can import into their copy of EDA.

4.3 Dashlet types

Dashlets are the building block from which you can build your dashboard. Several types of dashlets are available in EDA; each can be dragged and dropped on to your dashboard design. If you have added flex columns or rows, you can distribute dashlets within the resulting grid.

4.3.1 The counts dashlet

The counts dashlet displays a simple count of qualifying instances of something in EDA. You select a data source, and can then specify criteria to distinguish qualifying instances of the selected data that are counted and highlighted, versus the basic number of all records in the selected data source.

Figure 12: A sample counter dashlet

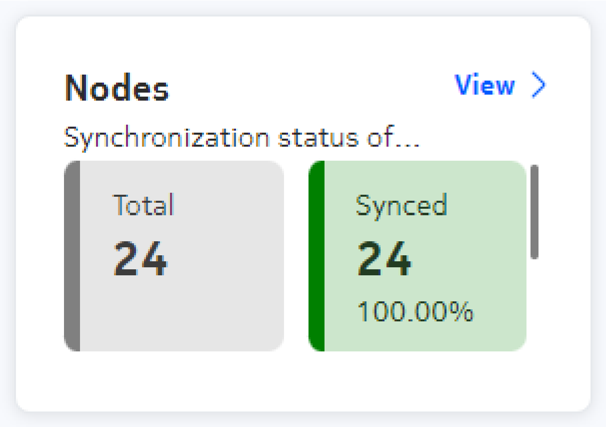


Table 8: Count dashlet properties

Property	Description
Common properties	
Title	The title of the dashlet when displayed in the EDA UI.
Subtitle	A subtitle, displayed below the title and in a smaller font.
Navigation target	Adds a 'View' button to the dashlet that links to the specified UI navigation target.
Fill available width	Dynamically changes the dashlet width based on the browser window and neighboring dashlets.
Dashlet width	The width of the dashlet.
Dashlet height	The height of the dashlet.
API Specification	
Query	Click the More icon to open a page on which to configure the data source for this dashlet. On that page you configure the data source as one of the following: **1.** EQL Query **2.** Natural Language query **3.** GVK Definition **4.** URL Endpoint
Counters	These properties configure the highlighting of values that meet criteria on the counts dashlet: • **Label**: the label shown beside qualifying values • **Color**: the color used to highlight qualifying values

Property	Description
	• **Field**: the field within the data source to be evaluated for possible highlighting • **Criteria** (Equals, Not Equal, Greater Than, Less Than): the logical operator that qualifies for this highlight (in combination with Value) • **Value**: the comparison value for the logical criterion.
Additional dashlet properties	
Show total	Indicates whether to display a count of all values retrieved in the source data set should be displayed on the chart, in addition to qualifying values.
Show total at end	When the total is shown, controls the position of the total display. Toggles between the total being the first count, or the last.
Show percentage	Indicates whether the counter should display what percentage of all values are represented by qualifying values.
Vertical lists	When the total is shown, controls the position of the total count and qualifying count. Toggles between the total being above, or below the count of qualifying values.

4.3.2 The line chart dashlet

A line chart dashlet places a line chart on the dashlet. It supports both stacked line charts (in which values are successively added to show a series of cumulative totals) and overlaid (a standard line chart in which values are displayed independently, not as a sum).

Figure 13: A sample line chart dashlet

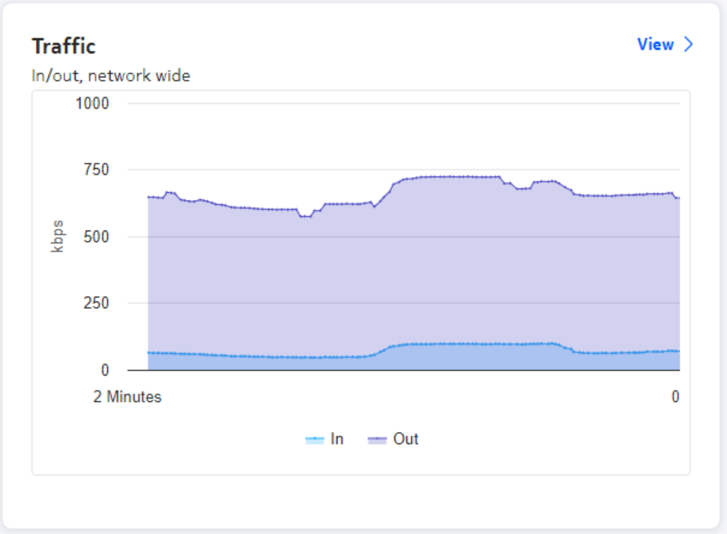


Table 9: Line chart dashlet properties

Property	Description
Common properties	
Title	The title of the dashlet when displayed in the EDA UI.
Subtitle	A subtitle, displayed below the title and in a smaller font.
Navigation target	Adds a 'View' button to the dashlet that links to the specified UI navigation target.
Fill available width	The width of the dashlet.
Dashlet width	When placed in a grid with flexcolumns, the number of adjacent columns this dashlet occupies.
Dashlet height	The height of the dashlet.
API Specification	
Query	Click the More icon to open a page on which to configure the data source for this dashlet. On that page you configure the data source as one of the following: **1.** EQL Query **2.** Natural Language query **3.** GVK Definition **4.** URL Endpoint
Chart Configuration	These properties control the display of the line chart: • Maximum number of data points • Y-Axis Units • Scaling Function (None, Metric Prefix Scaling)

4.3.3 The donut dashlet

A donut dashlet places a pie chart on the dashboard layout. You must configure a data source, and then set criteria for various pie slices describing qualifying subsets of that data. Many parameters are available to control the way the appearance of the chart and the individual pie slices.

Figure 14: A sample donut dashlet

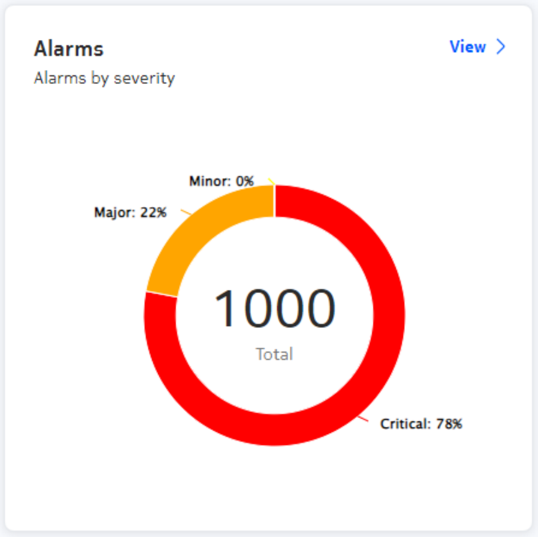


Table 10: Donut dashlet properties

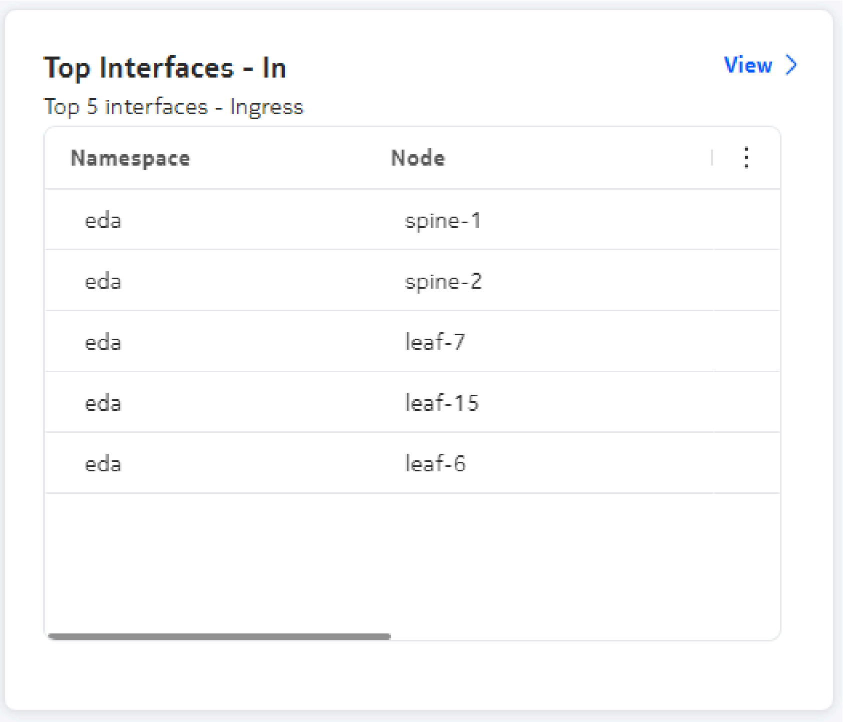
Property	Description
Common properties	
Title	The title of the dashlet when displayed in the EDA UI.
Subtitle	A subtitle, displayed below the title and in a smaller font.
Navigation target	Adds a 'View' button to the dashlet that links to the specified UI navigation target.
Fill available width	Dynamically changes the dashlet width based on the browser window and neighboring dashlets.
Dashlet width	The width of the dashlet.
Dashlet height	The height of the dashlet.
Charts	
Charts	A single pie chart dashlet can include multiple pie charts. Use this space to add and configure each pie chart. After configuring a pie chart, click the + icon to add and configure an additional pie chart for this dashlet.
Donut Chart Details An individual pie chart within the donut dashlet is configured on this page.	
Query Definition	Specifies the data source on which the pie chart's segments is based. Choose from: • EQL Query

Property	Description
	• Natural Language query • GVK Definition • URL Endpoint
Hide title	Indicates whether to show the chart title on the chart, or not. Toggles Yes or No.
Show total	Indicates whether the sum of all segments should be displayed on the chart, or not. Toggles Yes or No
Show slice labels	Indicates whether each chart segment should display a label for its data. Possible values: All, Percent, None
Slices: these properties control the display of each slices in the chart. Configure and add as many slices as your chart requires.	
Label	Indicates whether this slice should display its own label.
Color	The shading color applied to this slice.
Field	From the selected data source, the individual field corresponding to this slide.
Criteria	The logical criterion for this slide (Equals, Not Equal, Greater Than, Less Than)
Value	The fixed value against which the current field value and the Criteria are compared.
+	Click this icon to add the slice configuration to the set of slices included in this chart.

4.3.4 The data view dashlet

A data view dashlet places a data grid on the dashboard. You must specify a data source as part of the dashlet design..

Figure 15: A sample dataview dashlet



Top Interfaces - In

Top 5 interfaces - Ingress

Namespace	Node
eda	spine-1
eda	spine-2
eda	leaf-7
eda	leaf-15
eda	leaf-6

Table 11: Dataview dashlet properties

Property	Description
Common properties	
Title	The title of the dashlet when displayed in the EDA UI.
Subtitle	A subtitle, displayed below the title and in a smaller font.
Navigation target	Adds a 'View' button to the dashlet that links to the specified UI navigation target.
Fill available width	Dynamically changes the dashlet width based on the browser window and neighboring dashlets.
Dashlet width	The width of the dashlet.
Dashlet height	The height of the dashlet.
Charts	
Query	Click the More icon to open a page on which to configure the data source for this dashlet. On that page you configure the data source as one of the following: **1.** EQL Query **2.** Natural Language query

Property	Description
	3. GVK Definition **4.** URL Endpoint
Show information panel	Indicates whether an information panel should be available on this dashlet.
Show status bar	Indicates whether to include a status bar on the dashlet, showing (for example) whether any filters are applied, and the total number of rows in the list.

4.4 Using the dashboards list

Procedure

Step 1. Open the Dashboards page in the EDA GUI by selecting **Main** → **Dashboards**.

Expected outcome

The Dashboards list displays, showing all currently configured dashboards.

Note: By default, the list shows Custom dashboards. The Core dashboards list (selectable using a drop-down the navigation bar at the top of the page) is not currently enabled.

Step 2. Do any of the following:

- To preview an existing dashboard, click the **Table row actions** icon and select **Preview** from the displayed list.
- To edit an existing dashboard, click the **Table row actions** icon and select **Edit** from the displayed list.
- To delete an existing dashboard, click the **Table row actions** icon and select **Delete** from the displayed list, then click **OK** in the resulting confirmation dialog.

4.5 Designing a dashboard

About this task

This task guides you through the steps of adding and configuring layouts within a single dashboard by:

- creating the new dashboard
- adding a single layout: either a list layout, or a dashboard layout consisting of one or more dashlets
- configuring the data source for each list or dashlet, and configuring the appearance and behavior of each.
- optionally adding more list or dashboard layouts to the same dashboard
- saving your layout

Procedure

Step 1. Open the Dashboards page by selecting **Main** → **Dashboards** form the navigation menu.

Expected outcome

This opens the Dashboards page, showing a list of already-configured dashboards.

Step 2. Choose one of the following:

- To create a dashboard with a list layout, go to step 3.
- To create a dashboard with a set of dashlets, go to step 9.

Step 3. Use the drop-down control beside the **Create** button to select **Create List Layout**.

Step 4. In the **Definition** panel, configure basic properties for the dashboard:

- **Layout name**: an internal name for this layout. This name cannot include spaces or special characters.
- **Display name**: the name for this layout, as displayed within the EDA GUI. Unlike the layout name, you can include spaces in the Display name.
- **Description**: an optional description of the layout and its purpose.
- **Show navigation toolbar?**: governs whether a breadcrumb bar displays above the dashboard.

Step 5. Click on the list in the center configuration panel to reveal properties for the list in the Properties panel.

Step 6. Configure display properties for the list:

- **Show information panel**
- **Show status bar**
- **Show column filters**

Step 7. Configure the source data for the list:

a. Click the vertical dots icon beside the **Query** field to open a window in which to configure data source for the list.

b. In the Data View Details window, use the first drop-down to select a source type for the dashlet's query:

- EQL Query
- Natural Language query
- GVK Definition
- URL Endpoint

c. Use the second field to enter the query expression, or to specify the GVK definition or URL endpoint.

d. Click **Query** to retrieve data associated with the expression you entered.

e. Click **Save**.

Step 8. Go to step 19

Step 9. Click **Create**.

Step 10. In the **Definition** panel, configure basic properties for the dashboard:

- **Layout name**: an internal name for this layout. This name cannot include spaces or special characters.

- **Display name**: the name for this layout, as displayed within the EDA GUI. Unlike the layout name, you can include spaces in the Display name.
- **Description**: an optional description of the layout and its purpose.
- **Show navigation toolbar?**: ADD DEFINITION

Step 11. In the **Properties** panel, configure screen properties for the dashboard:Screen Name:

- **Screen Name**: DEFINE
- **Screen Type**: This is set to Dashboard and cannot be altered.
- **Show screen in Navigation Bar?**:
- **Default screen when layout is loaded?**:

Step 12. Click the **Library** tab to configure the dashboard layout.

Step 13. Optionally, add rows and columns to the dashboard:

Note: You can arrange dashlets on the dashboard even without creating rows and columns in advance; but configuring a grid gives you more control over dashlet positioning, and allows you to configure dashlets to span multiple cells in the grid arrangement.

a. To add rows to the dashboard, drag the **Flex Row** control into the layout panel. Repeat this to add more rows to the dashboard.

b. To add columns to the dashboard, drag the **Flex Column** control into the layout panel. Repeat this to add more columns to the dashboard.

c. In the Properties panel, configure the flex row or flex column you added by setting the **Vertical Alignment** and **Horizontal Alignment** properties.

Step 14. Add a dashlet to the dashboard by selecting a **Dashlet** control from those displayed, and dragging it into the layout area. If you previously added rows or columns, drop the dashlet into the appropriate position.

Step 15. Click on the dashlet in the center configuration panel to reveal properties for the dashlet in the Properties panel.

Step 16. Configure the dashlet by setting:

- Screen properties (these are common to all dashlets).
- Dashlet properties (some are common to all dashlets; others vary by dashlet type).

Note: See the topics for dashlet types for details about the individual parameters available for each type of dashlet.

Step 17. To configure the source data for the dashlet (among the dashlet properties):

a. Click the vertical dots icon beside the **EQL Query** field to open a window in which to configure data source for the dashlet.

b. In the Details window, use the first drop-down to select a source type for the dashlet's query:

- EQL Query
- Natural Language query
- GVK Definition

   - URL Endpoint

c. Use the second field to enter the query expression, or to specify the GVK definition or URL endpoint.

d. Click **Query** to retrieve data associated with the expression you entered.

e. Click **Save**.

f. Configure additional properties for the data, if they are available for your dashlet type.

> **Note:** For example, a Counter dashlet allows you to specify here whether the counter should display a total, total at end, percentage, or a vertical list of values.

Step 18. Repeat steps 14, 16 and 17 to add more dashlets to the dashboard if required, until all dashlets are configured.

Step 19. Do any of the following:

- To save your dashboard, click the **Save** icon.
- To add a new dashboard layout to your dashboard, click **Add**.
- To add a new list layout to your dashboard, use the drop-down beside the **Add** control to select **Add list Layout**.
- To preview your dashboard, click the **More** icon and select **Preview Saved Changes** from the list of actions.
- To save your dashboard layout as a file, suitable for others to import into their EDA system, click the **More** icon and select **Export** from the list of actions.

5 Namespaces

A namespace is a logical partition within a cluster that provides a mechanism for isolating sets of resources from each other. Such resource segmentation allows multiple teams or applications to share the same cluster without conflict, because each has its own set of resources in its own namespace.

Using namespaces, you can use a single EDA instance to manage multiple sets of resources with distinct administrators. Each EDA user can also be granted access to only selected namespaces. A common real-world case for such system is an operator with regional operations teams, where a single controller instance supports all of the regions, but users within a region can only see the resources and states relating to their region.

The base namespace

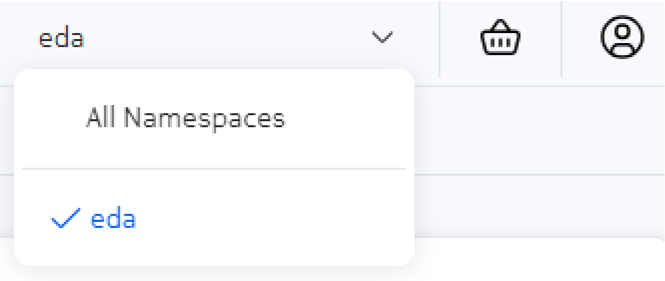
EDA always includes one built-in namespace; by default, this is `eda-system`. The default can be modified during EDA installation.

All EDA core services run in the base namespace, including Pods for NPP and CX-simulated TopoNodes. Unnamespaced resources (those with "namespaced" set to false in their manifest) also exist in the base namespace.

Namespaces in the EDA GUI

The top of every page in the EDA GUI includes a namespace selector. You must use to specify the namespace you are working in.

Figure 16: Namespace selector

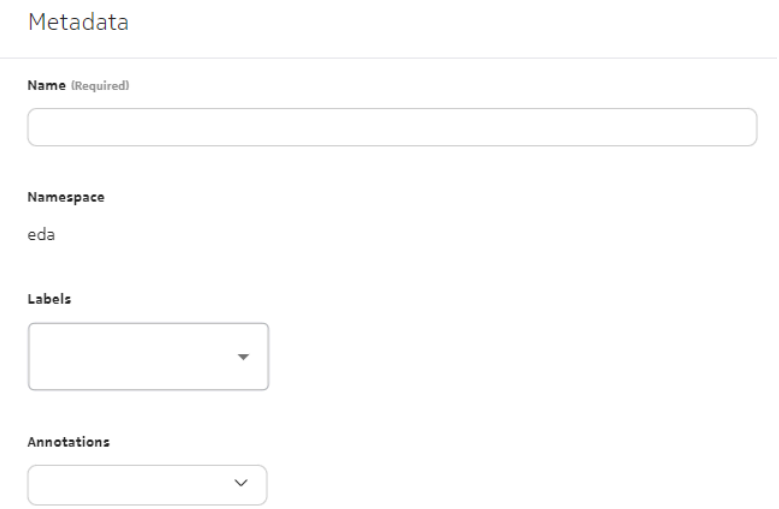


The data displayed in data grids always conforms to the selected namespace.

- If you have selected All Namespaces, then data grids contain data from all namespaces. You must have permissions defined in a Cluster Role to access data in this view.
- If you have selected a specific namespace, then data grids contain data exclusively from that namespace. You must have permissions defined in either a Cluster Role or a namespace Role to access data in this view.

The currently selected namespace is automatically used as the **Namespace** value for any resource you create in the GUI. To create a resource in a different namespace, you must select the intended namespace in the selector.

Figure 17: Namespace is read-only metadata derived from the current namespace selection



Metadata

Name (Required)

Namespace

eda

Labels

Annotations

User-created namespaces

You can create new namespaces using the EDA GUI.

5.1 Creating a namespace

About this task

Only users with sufficient privileges can create a new namespace.

Procedure

Step 1. Open the **System Administration** menu.

Step 2. Click **Namespaces** to open the Namespaces page.

Step 3. On the Namespaces page, click **Create**.

Step 4. Enter the following details about the new namespace:

- The **Name** of the namespace
- One or more **Labels**
- One or more **Annotations**
- An optional **Description** for the Namespace

Step 5. Commit the new namespace by doing one of the following:

- Click **Commit** to commit the new configuration immediately.
- Click **Add to Transaction** to add this configuration to a cumulative transaction.

6 Resources

In EDA, a resource is a unit of automation and can represent virtually anything:

- an interface on a network device
- a complete fabric configuration
- a network service like a VPN or a VRF
- and even non-network related resources like a user account, a DNS record, or a firewall rule.

As a Kubernetes citizen, EDA represents its resources using Custom Resources (CRs) of Kubernetes that can be created using multiple methods including the Kubernetes (K8s) API, the EDA API, or through a User Interface (UI). By using CRs, EDA also implements the Kubernetes Resource Model, or KRM.

The KRM defines how Kubernetes resources are described, created, updated, and monitored. Kubernetes resources consist of a combination of fields that describe their state and behavior within the cluster, most importantly the `spec`, `status`, and `metadata` fields.

In Kubernetes, a resource is any object the Kubernetes API can create and manage. These resources represent various entities, such as `Pods`, `Services`, `Deployments`, `ConfigMaps`and so on., which are essential for orchestrating containerized applications.

Every resource in Kubernetes is defined using a standard structure that includes `metadata`, a `spec`, and a `status`. Where:

- `metadata` provides unique identifiers and metadata for resources.
- `spec` provides the specification for the resource - its configuration.
- `status` provides an interface for the controller/resource to publish relevant information back to the user/operator.

Derived resources

As part of the execution of a transaction, EDA applications sometimes generate a set of resources. These resources are not "owned" by the user or operator; instead, they owned by the application that generated them. To ensure the ongoing operation of the owning application, such resources can only be changed by that same application.

In EDA such a resource is known as a derived resource; it is a resource whose entire content is derived from some other resource.

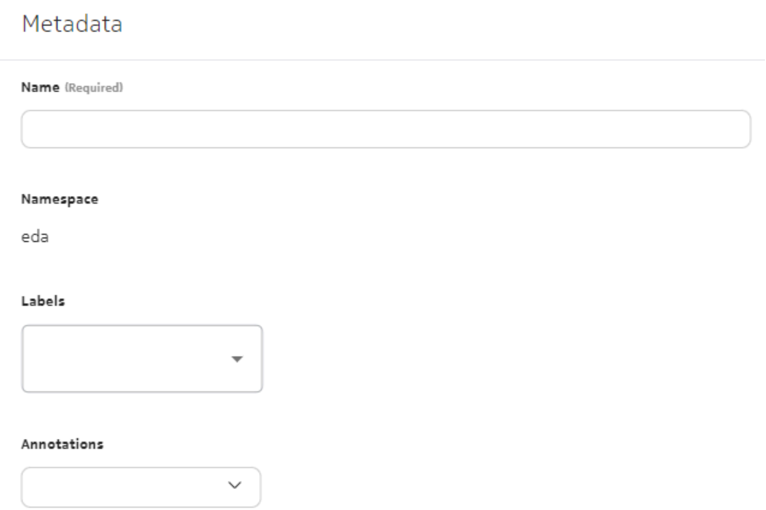
The EDA GUI prevents you from modifying or deleting derived resources. To indicate that a resource is derived and cannot be modified or deleted, derived resources are presented as read-only, and the usual modification actions are restricted; for example, EDA does not allow you to use a Delete action to delete a derived resource. Unavailable actions are grayed out in action lists.

In data grids, rows displaying derived resources are shaded to indicate that those resources cannot be modified or deleted.

6.1 Labels

EDA uses labels to organize and describe resources. Labels are among the metadata common to all resources in EDA. In the EDA GUI, labels can be viewed and entered in the Metadata panel for a resource.

Figure 18: Resource metadata



Metadata

Name (Required)

Namespace

eda

Labels

Annotations

Labels are not mere descriptions of objects; they are also used throughout EDA as the basis for selecting objects. You can apply the same label to a set of objects and then manipulate them as a group based on that shared label. This makes it easier for system administrators and operators to manage large-scale clusters.

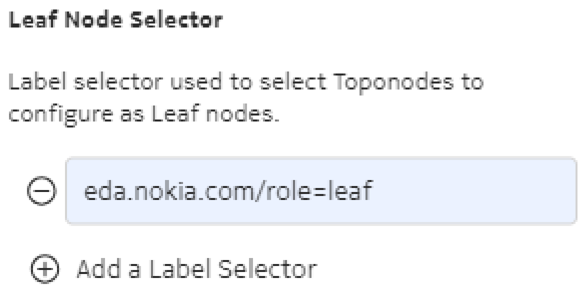
A label consists of two pieces of information: a key, and a value. Labels are limited to key-value pairs of small size and are designed for simple, static values. For example:

- app=frontend
- version=v1.0
- environment=prod

The key can include up to 253 characters if using the DNS subdomain format (`<domain>/<key>=<value>`), and the value can include up to 63 characters.

Labels are particularly useful for selecting objects; for example, you can use a label to indicate which pods a service should treat as traffic destinations. The following illustration shows a segment of a fabric configuration in which participating leaf nodes are selected among those that possess the label "role" and its value is "leaf". Additional labels can be selected to narrow down the set of qualifying nodes.

Figure 19: Selecting objects based on labels



Leaf Node Selector

Label selector used to select Toponodes to configure as Leaf nodes.

eda.nokia.com/role=leaf

Users and application writers can:

- apply labels arbitrarily to resources
- select resources within their application based on these labels

Label changes are considered normal changes for the purposes of transactions. A label change can trigger execution of scripts, and if executions are successful their changes are persisted to git.

Labels are a flexible way to decouple the interactions between resources, but they do have some limitations. In particular, the value of a label is limited to 63 characters, and Kubernetes resource names are limited to 253 characters. This means that labels cannot reliably encode a resource name, for example.

Note: Labels in EDA work slightly differently from labels in Kubernetes. EDA still stores labels in the metadata of a resource as does Kubernetes, but the means by which you select based on labels is slightly different. In particular, Kubernetes objects typically use the `metav1.LabelSelector` Go struct in order to select labels of a certain resource type. This `LabelSelector` is not supported in EDA.
Instead, EDA uses one or more string expressions to select. An expression can contain one or more selectors, separated by `,`. Selectors are AND'd together, similar to Kubernetes' `LabelSelector`. A selector supports various operators, including but not limited to `=`, `!=`, `in`, `notin`.

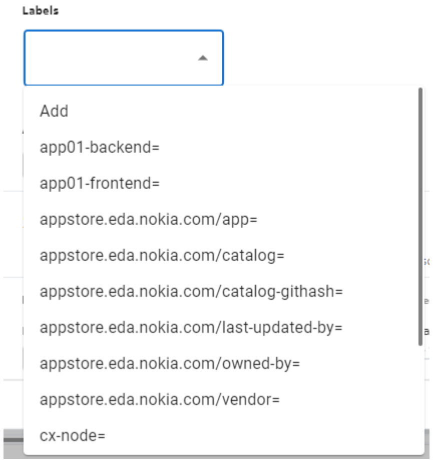
Some examples:

- `app=cat` means a resource is only returned if it has a label present named `app`, with a value of `cat`.
- `app in (cat)` is another way of writing the above, meaning a resource is only returned if it has a label named `app` with a value of `cat`.
- `app` returns a resource if it has a label present with the name `app`, with any value (including an empty value).
- `!app` returns a resource if it does not have a label present with the name `app`, with or without a value.
- `app in (cat, dog)` returns a resource if it has a label present with the name `app`, with a value of `cat` OR `dog`.
- `app in (cat, dog),env in (prod, demo)` returns a resource if it has both a label named `app` with values `cat` OR `dog`, AND a label named `env` with values `prod` OR `demo`.
- `app notin (elephant, rhino)` returns a resource if it does NOT contain a label named `app` with a value of either `elephant` OR `rhino`.

- `app=cat,env=prod` returns a resource if it has a label named `app` with the value `cat`, AND a label named `env` with value `prod`.

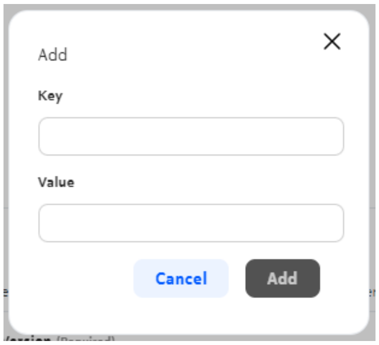
Selecting or creating a label

In the EDA GUI, where a **Label** field is present, you can enter a label by clicking in the **Label** field. This displays a list of available labels to choose from:


Labels
Add
app01-backend=
app01-frontend=
appstore.eda.nokia.com/app=
appstore.eda.nokia.com/catalog=
appstore.eda.nokia.com/catalog-githash=
appstore.eda.nokia.com/last-updated-by=
appstore.eda.nokia.com/owned-by=
appstore.eda.nokia.com/vendor=
cx-node=

To use an existing label, select it in the list. To narrow the list of displayed labels, type the first few letters of a label you are looking for; the list filters to show only the labels that match the text provided.

To create a new label, click **Add** to open the label creation window:

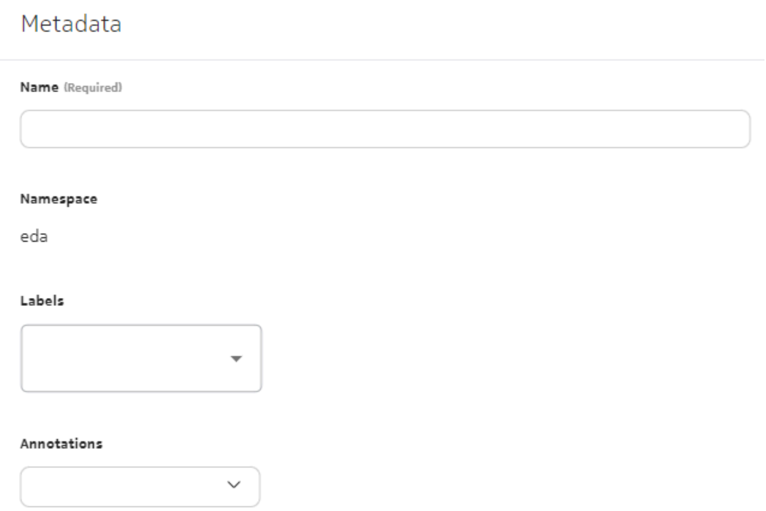


Enter a **Key** and a **Value**, then click **Add**.

6.2 Annotations

EDA uses annotations to organize and describe resources. Annotations are among the metadata common to all resources in EDA. In the EDA GUI, annotations can be viewed and entered in the Metadata panel for a resource.

Figure 20: Resource metadata



Metadata

Name (Required)

Namespace

eda

Labels

Annotations

Annotations are similar to labels, but are used for different purposes.

Like a label, an annotation consists of a key and a value. However, annotations values are not subject to the same length restrictions as labels. Annotations can store lengthy information that resembles the information contained in labels, but frequently overruns labels length restrictions.

Like labels, annotations are metadata about an object. But unlike labels, annotations do not influence the system’s behavior. Annotations are not used for selection or querying. They are not indexed and do not affect any selection logic. Annotations are more informational; and although they are not used by EDA's resource selection systems, they can still be useful to external systems, people, or automation tools.

Annotations are typically used to store arbitrary, unstructured data like configuration details, URLs, object tracking information, or any other information that does not need to be part of Kubernetes’ logic. They are useful for attaching large or complex data that doesn’t need to be indexed, like CI/CD metadata, deployment signatures, or documentation links.

The EDA system uses annotations to store two types of data:

- ConfigEngine uses the annotations property to tag resources for which transactions have failed.
  The system-generated annotation text indicates that the resource is part of a failed transaction, and the Kubernetes-visible version of the resource may not be aligned with the running/actual version.
- The system uses the annotations property to store resource names
  This is primarily used with derived resources, where it is useful to be able to see the hierarchy of resources - for example a `VirtualNetwork` generating a `BridgeDomain`.

Examples of possible annotations values:

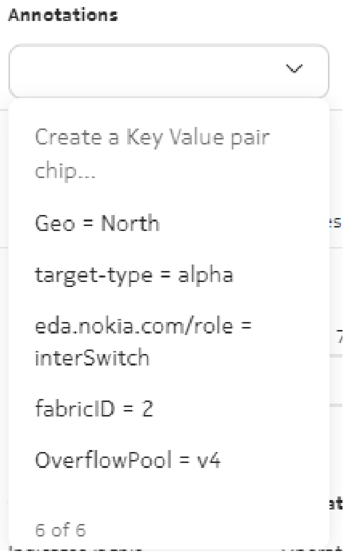
- kubectl.kubernetes.io/last-applied-configuration="JSON"
- author=team-name
- description="Stores the last applied configuration of a resource for use by kubectl apply"

Annotation changes are considered normal changes for the purposes of transactions. They trigger execution of scripts, and if executions are successful their changes are persisted to git. However there are a small number of exceptions. EDA does not trigger, monitor, or persist any annotations with the following keys:

- `core.eda.nokia.com/failed-transaction`
- `core.eda.nokia.com/running-version`
- `kubectl.kubernetes.io/last-applied-configuration`

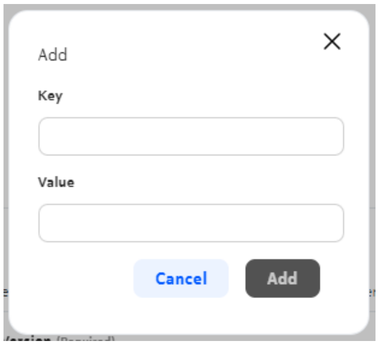
Selecting or creating an annotation

In the EDA GUI, where an **Annotation** field is present, you can enter an annotation by clicking in the **Annotation** field. This displays a list of available annotations to choose from:



To use an existing annotation, select it in the list. To narrow the list of displayed annotations, type the first few letters of a label you are looking for; the list filters to show only the annotations that match the text provided.

To create a new annotation, click **Create a Key Value pair chip...** to open the annotation creation window:



Enter a **Key** and a **Value**, then click **Add**.

7 Workflows and workflow definitions

A workflow is a sequence of steps required to perform some process.

The concept of a workflow is typically used in automation platforms to make an operational task reproducible; it is the logic or code required to execute the task. It is not too much of an extrapolation from the concept of a workflow to a CI/CD pipeline, which describes a sequence of actions to run, with workflow semantics between them.

In EDA, workflows can define the steps required to upgrade a `TopoNode`, validate connectivity in a `VirtualNetwork`, or perform a simple ping operation.

A workflow in EDA is implemented via an orchestrated Kubernetes `Job`. Workflows themselves are container images that take some input, perform some work, and provide some output.

Before running any workflows, the relevant Workflow Definitions must be loaded into the system.

Workflow definitions

A `WorkflowDefinition` in EDA binds together two things:

- a container image whose entrypoint executes a workflow
- a name in the system

A workflow definition also allows its creator to define the input of the workflow, and its output.

A workflow definition can include the following fields:

- The usual metadata, including:
  - `Name`
  - `Labels`
  - `Annotations`
- `image`: indicates a path to an image to use when executing this workflow.
- `imagePullSecrets`: the secrets required to pull the image for this workflow
- `jsonSchemaSpec`: a full JSON schema defining the input the workflow accepts.
- `jsonSchemaStatus`: a full JSON schema defining the output this workflow populates.

In EDA, both workflow and CI/CD functionality are supported through the WorkflowEngine; the controller behind the instantiation, status, and interaction with the `Workflow` and `WorkflowDefinition` resources.

Sample workflow definitions

Environment CLI plugin workflow definition

```
apiVersion: core.eda.nokia.com/v1
kind: WorkflowDefinition
metadata:
  labels:
    eda.nokia.com/source: derived
  name: environment-cliplugin
  annotations: {}
spec:
  image: >-
    <registry>/sr/eda/scripts/workflow-environment-cliplugin:1.0.0
  imagePullSecrets:
    - <secrets>
  flowDefinitionSchema:
    jsonSchemaSpec: ''
    jsonSchemaStatus: ''
status: {}
```

Services Edge Ping workflow definition

```
apiVersion: core.eda.nokia.com/v1
kind: WorkflowDefinition
metadata:
  labels:
    eda.nokia.com/source: derived
  name: services-edgeping
  annotations: {}
spec:
  image: >-
    <registry>/sr/eda/scripts/workflow-services-edgeping:1.0.0
  imagePullSecrets:
    - <secrets>
  flowDefinitionSchema:
    jsonSchemaSpec: ''
    jsonSchemaStatus: ''
status: {}
```

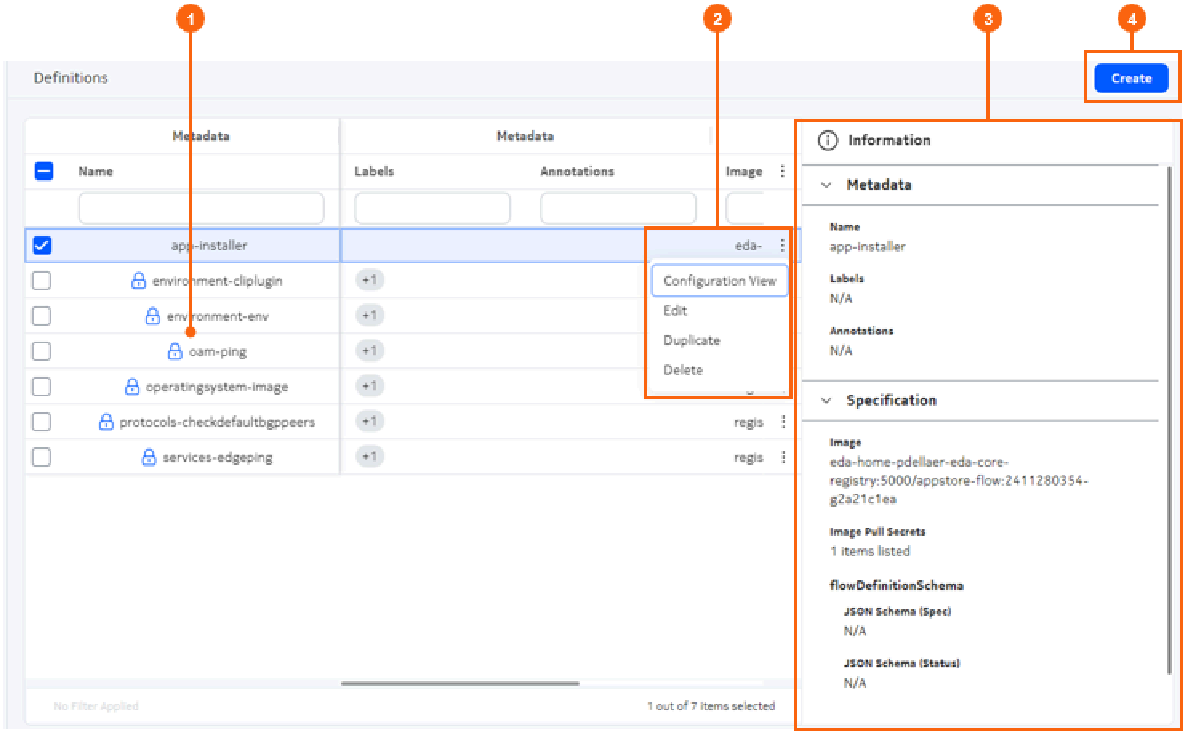
7.1 The workflow definitions page

The Workflow Definitions page in the EDA GUI allows you to view, delete, and sometimes edit and duplicate those workflow definitions that are known to EDA.

Note: For workflow definitions that are installed (and owned) by apps, only the View configuration and Delete actions are available. Duplicating and editing these workflow definitions is not permitted.

On this page you can also create new workflow definitions. Workflow definitions are a normal EDA resource and are defined by a Custom Resource Definition component of a Manifest, whose creation or modification you commit or add to a larger transaction like any other EDA resource.

Figure 21: The workflow definitions page



	Metadata	Metadata		
☐	Name	Labels	Annotations	Image
☑	app-installer			eda-
☐	environment-cliplugin	+1		
☐	environment-env	+1		
☐	oam-ping	+1		
☐	operatingsystem-image	+1		
☐	protocols-checkdefaultbgppeers	+1		regis
☐	services-edgeping	+1		regis

No Filter Applied

1 out of 7 items selected

Name
app-installer

Labels
N/A

Annotations
N/A

Image
eda-home-pdellaer-eda-core-registry:5000/appstore-flow:2411280354-g2a21c1ea

Image Pull Secrets
1 items listed

flowDefinitionSchema

JSON Schema (Spec)
N/A

JSON Schema (Status)
N/A

Table 12: Elements of the Workflows page

#	Name	Function
1	Workflow definitions list	A list of workflow definitions known to EDA. Some of these workflow definitions may be created by users. Other, indicated by a lock icon, were added to EDA when specific applications were installed.
2	Actions menu	Displayed when clicking on the **More** icon at the right edge of the row for an individual workflow definition. Available actions are: • **Configuration view**: opens the workflow definition view, displaying details of its metadata, specification, schema, and status. The view is read-only. • **Edit**: like Configuration view, this opens the workflow definition view, but the configuration can be modified and any changes committed. This option is not available for workflow definitions owned by individual EDA applications. • **Duplicate**: allows you to save a copy of the selected workflow definition with a new name. This option is not available for workflow definitions owned by individual EDA applications.

#	Name	Function
		- **Delete**: Deletes the selected workflow definition, after confirmation.
	Information panel	When expanded, the information panel displays details about the selected workflow definition.
	Create button	Click to open the Definitions page to create and commit a new workflow definition.

8 Alarms

An alarm is an unexpected condition in EDA or the EDA-managed network that can be cleared by attaining some corrective state. The primary purpose of an alarm is to bring some abnormal condition to the attention of an operator, and thereby support debugging and resolution.

Note: An alarm is distinct from an event. An alarm represents a condition that should be corrected by reaching some associated corrective state. An event is a one-time occurrence and message (typically a log entry) about something that has occurred in the system. No future state is expected to clear an event, and it does not require an action to clear. Alarms are stateful; events are not.

Alarms in EDA can arise from a variety of sources, including the EDA system itself and the wide array of supported apps. For any alarm, the source/affected object is identified as part of the alarm in the `group` and `kind` fields.

Alarm are also associated with a namespace; this could be the base EDA namespace, or some other namespace. Users can only see and interact with alarms in namespaces for which they have access permissions.

Some alarms can be generated by intent-based apps within EDA. EDA treats such alarms as having been cleared if the app stops reporting that alarm.

Alarms associated with apps are described in documentation for individual apps.

Alarms on standby clusters

Standby cluster alarms can be important in understanding the state of redundancy in an EDA cluster. It is therefore useful to be able to see alarms generated on a standby cluster member even when working with the active member.

EDA supports this using the \`cluster_member\` field, which is set to the name of the cluster member that raised the alarm. This allows an operator to view alarms for all clusters, but still distinguish alarms for the active cluster from those for a standby cluster. For alarms that are not cluster-specific, this field remains unset.

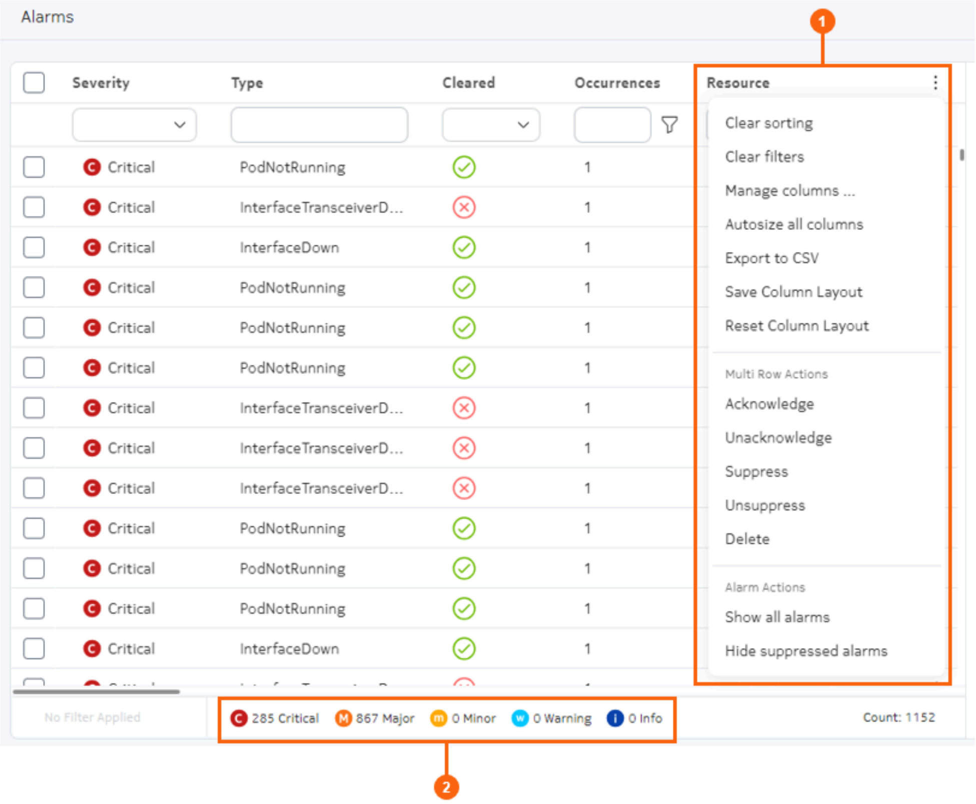
Alarms in the EDA GUI

An alarm summary is displayed on the EDA home page. Beyond that summary, alarms are primarily displayed, and interacted with, on the Alarms page. There you can do the following for individual alarms, or as a bulk operation to a number of concurrently selected alarms:

- Suppress an alarm: this sets the suppressed flag for the current instance of the alarm. By default, suppressed alarms are not displayed in the EDA GUI.
- Delete an alarm: this removes all history of the alarm. Deletion is only allowed for cleared alarms. The option is disabled for active alarms.
- Acknowledge an alarm: this sets the Acknowledged flag for the current instance of the alarm.

8.1 The Alarms page

Figure 22: The EDA Alarms page



Alarms

Severity	Type	Cleared	Occurrences	Resource
Critical	PodNotRunning	✓	1	
Critical	InterfaceTransceiverD...	✗	1	
Critical	InterfaceDown	✓	1	
Critical	PodNotRunning	✓	1	
Critical	PodNotRunning	✓	1	
Critical	PodNotRunning	✓	1	
Critical	InterfaceTransceiverD...	✗	1	
Critical	InterfaceTransceiverD...	✗	1	
Critical	InterfaceTransceiverD...	✗	1	
Critical	PodNotRunning	✓	1	
Critical	PodNotRunning	✓	1	
Critical	PodNotRunning	✓	1	
Critical	InterfaceDown	✓	1	

No Filter Applied | 285 Critical | 867 Major | 0 Minor | 0 Warning | 0 Info | Count: 1152

Table 13: Elements of the Alarms page

#	Name	Function
1	Alarms menu	The Alarms menu includes: • common table controls • multi-row actions unique to alarms • special actions unique to the alarms list
2	Alarm count	Displays the number of current alarms of various severities.

Columns

The list of alarms displays the following columns by default.

Table 14: Alarm display columns

Column	Description
Severity	The importance of the alarm, as defined by the alarm itself. Supported severities are: • Critical • Major • Minor • Warning • Info
Type	The alarm type, as defined by the alarm itself. For example, InterfaceDown.
Cleared	Whether the alarm has been cleared by an operator. Possible values are: • True • False
Occurrences	The number of occurrences for the alarm.
Resource	Indicates the name of the resource that this alarm is present on. • For a TopoNode target this is the TopoNode name, for example "spine-1-1". • For non- `TopoNode` target, this may be the name of a Pod. • A resource may also be an instance of a resource - for example if an alarm was raised against a Fabric, the endpoint would be set to the name of the specific instance of a Fabric the alarm was raised against.
Kind	Indicates the kind of resource the alarm is present on. • Most commonly this is set to TopoNode, but may be a component within the EDA core - for example ConfigEngine or StateEngine. • This is set to Fabric if an alarm is raised against a Fabric.
Group	Indicates the group of the resource the alarm is present on. • Most commonly this is set to core.eda.nokia.com, , but may be any other app-provided group. • This is set to fabrics.eda.nokia.com if an alarm is raised against a Fabric.
Last Changed	Indicates the time the alarm last changed state. The timestamp is updated any time an alarm changes state between cleared and not cleared.

Related topics

Working with data grids

Namespaces

8.2 Sample core alarms

Table 15: Repository Reachability Down

Property	Description
Name	RepositoryReachabilityDown-<cluster>-<server-name>-<repo-type>-<source>
Severity	Critical
Description	Connectivity between <source-kind> "<source>" and the "<repo-type>" repository at "<server-uri/remote-path>" is down. This alarm is raised after three failures to connect to a repository, where each attempt is made at a 15s interval. After three failures the alarm is generated (so after 45s), and is cleared on a connection attempt succeeding.
Probable cause	Connectivity issues, Kubernetes CNI misconfiguration, or credential/TLS misconfiguration/expiration.
Remedial action	Restore connectivity between the corresponding <source-kind> and apps repository/git server. Ensure credentials and proxy configuration is correct, and any offered certificates are trusted.
• server-name is the name of the Git server hosting the repository, for example primary. • repo-type is the type of repository, one of Apps, Backup, Identity, Security, Catalog, UserSettings. • source is the name of the pod that raised the alarm, for example eda-se-1. • source-kind is one of ConfigEngine, StateEngine, AppStore. • cluster is the name of the cluster member that this alarm was raised in, populated via setting the clusterSpecific flag. • server-uri/remote-path is the combination of the server URI and remote path as defined in the Engine Config.	

Table 16: Service Reachability Down

Property	Description
Name	ServiceReachabilityDown-<cluster>-<service>-<source>
Severity	Critical
Description	Connectivity between <source-kind> "<source>" and the <kind> on "<service>" is down.
Probable cause	Connectivity issues between worker nodes in the Kubernetes cluster, Kubernetes CNI misconfiguration, pod failure, or TLS misconfiguration/expiration.
Remedial action	Restore connectivity between the corresponding source and destination. Ensure credentials and proxy configuration is correct (typically using no proxy for inter-cluster HTTPS), and certificate validity.

<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td colspan="2">• service is the common name of the destination pod, for example eda-npp-leaf-1-1. Engineering to comment if this is feasible.
• source is the name of the source pod that raised the alarm, for example eda-sc-1.
• source-kind is one of APIServer, ConfigEngine, StateEngine, AppStore.
• cluster is the name of the cluster member that this alarm was raised in, populated via setting the clusterSpecific flag.</td></tr>
</table>

Table 17: Pod Not Running

<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>Name</td><td>PodNotRunning-<cluster>-&lt;pod&gt;</td></tr>
<tr><td>Severity</td><td>Critical</td></tr>
<tr><td>Description</td><td>Pod "&lt;pod>" is not in the "Running" state. Any functionality provided by this pod is not available. This alarm can be raised transiently at system startup.</td></tr>
<tr><td>Probable cause</td><td>Kubernetes controller or registry reachability issues, worker node failure, initial instantiation.</td></tr>
<tr><td>Remedial action</td><td>Validate reachability to the registry used to pull the image for the specified pod, ensure no worker node, storage, or networking issues exist that would cause the Kubernetes controller to mark the pod in any state other than "Running".</td></tr>
<tr><td colspan="2">• cluster is the name of the cluster member that this alarm was raised in, populated via setting the clusterSpecific flag.
• pod is set to the name of the pod that is not running, but should be.
• config-engine is set to the name of the ConfigEngine pod that raised the alarm.</td></tr>
</table>

Table 18: Deployment Degraded

<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>Name</td><td>DeploymentDegraded-<cluster>-<deployment&gt;</td></tr>
<tr><td>Severity</td><td>Critical</td></tr>
<tr><td>Description</td><td>Deployment "<deployment>" has at least one replica not in the "Running" state. Depending on the application this may result in loss of functionality or loss of service capacity. This alarm can be raised transiently at system startup.</td></tr>
<tr><td>Probable cause</td><td>Kubernetes infrastructure issues, worker node failure, initial instantiation.</td></tr>
<tr><td>Remedial action</td><td>Validate reachability to the registry used to pull images for any failed pods in the Deployment, ensure no worker node, storage, or networking issues exist that would cause the Kubernetes controller to mark pods in any state other than "Running".</td></tr>
<tr><td colspan="2">• cluster is the name of the cluster member that this alarm was raised in, populated via setting the clusterSpecific flag.</td></tr>
</table>

Property	Description
• pod is set to the name of the pod that is not running, but should be. • config-engine is set to the name of the ConfigEngine pod that raised the alarm.	

Table 19: Deployment Down

Property	Description
Name	DeploymentDown-<cluster>-<deployment>
Severity	Critical
Description	Deployment "<deployment>" is down, with no pods in the "Running" state. Any functionality provided by the Deployment is not available. This alarm can be raised transiently at system startup.
Probable cause	Kubernetes infrastructure issues, worker node failure, initial instantiation.
Remedial action	Validate reachability to the registry used to pull images for failed pods in the Deployment, ensure no worker node, storage, or networking issues exist that would cause the Kubernetes controller to mark pods in any state other than "Running".
• cluster is the name of the cluster member that this alarm was raised in, populated via setting the clusterSpecific flag. • pod is set to the name of the pod that is not running, but should be. • config-engine is set to the name of the ConfigEngine pod that raised the alarm.	

Table 20: NPP Down

Property	Description
Name	PPDown-<cluster>-<npp>
Severity	Critical
Description	Connectivity between ConfigEngine "<config-engine>" and the NPP "<npp>" is down. This results in no new transactions succeeding to targets served by this NPP (unless operating in null mode), and no telemetry updates being received. Effectively targets served by this NPP are offline. Look for a corresponding PodNot Running alarm.
Probable cause	Connectivity issues between worker nodes in the Kubernetes cluster, Kubernetes CNI misconfiguration, pod failure, or TLS misconfiguration/expiration.
Remedial action	Restore connectivity between the corresponding ConfigEngine and the destination NPP. Ensure credentials and proxy configuration is correct (typically using no proxy for inter-cluster HTTPS), and certificate validity.
• cluster is the name of the cluster member that this alarm was raised in, populated via setting the clusterSpecific flag. • npp is set to the name of the destination NPP.	

Property	Description
• config-engine is set to the Pod name of the ConfigEngine that raised the alarm.	

Table 21: Pool Threshold Exceeded

Property	Description
Name	PoolThresholdExceeded-<pool-type>-<pool-name>-<pool-instance>
Severity	Varies; see definitions
Description	The "<pool-instance>" instance of the <pool-type> "<pool-name>" has crossed the <severity> threshold of <threshold>.
Probable cause	Pool utilization.
Remedial action	Expand the pool via growing a segment, or add additional segments. Additionally you may move pool consumers to a different pool.
• pool-type is the pool type, one of Subnet, IPInSubnet, Index, IP. • pool-name is the name of the allocation pool, for example ipv4-pool. • pool-instance is the name of the instance of the pool, for example global. • severity is the severity of the alarm, which increases based on which threshold has been breached. • threshold is the value of the threshold that has been breached, for example 80. • config-engine is the pod name of the ConfigEngine raising the alarm.	

Table 22: State Engine Reachability Down

Property	Description
Name	StateEngineReachabilityDown-<state-engine>-<state-controller>
Severity	Critical
Description	Connectivity between State Controller "<state-controller>" and the State Engine "<state-engine>" is down. This results in no new state application instances being deployed to the corresponding State Engine, and the rebalancing of already-pinned instances to other State Engines. This connectivity is also used to distribute the map of shards to State Engine, meaning the corresponding State Engine will not receive shard updates (assuming it is still running).
Probable cause	Connectivity issues between worker nodes in the Kubernetes cluster, Kubernetes CNI misconfiguration, pod failure, or TLS misconfiguration/expiration.
Remedial action	Restore connectivity between the corresponding State Controller and the destination State Engine. Ensure credentials and proxy configuration is correct (typically using no proxy for inter-cluster HTTPS), and certificate validity.
• state-engine is the name of the State Engine with connectivity issues, for example eda-se-1. • state-controller is the name of the State Controller pod that raised the alarm, for example eda-sc-1.	

8.3 Viewing alarms

About this task

The page in the EDA UI in which to view and interact with alarms is located at **Main** → **Alarms**.

Note: By default, the alarm list:

- is sorted first by "Severity", and then by the "last changed" timestamp in descending order (most recent change first)
- hides any suppressed alarms

Procedure

Step 1. Use standard EDA table controls to manage the alarms list. XREF

Step 2. To include suppressed alarms (which are hidden by default), do the following:

a. Click the **More** icon at the upper right of the Alarms page.

b. Select **Show All Alarms** from the displayed list.

Step 3. To exclude suppressed alarms from the list, do the following:

a. Click the **More** icon at the upper right of the Alarms page.

b. Select **Hide suppressed alarms** from the displayed list.

8.4 Acknowledging an alarm

Procedure

Step 1. Find the alarm in the list using the sorting and filtering controls.

Step 2. At the right side of the row, click the **Table row actions** button.

Step 3. Select **Acknowledge** from the list.

Step 4. Click **Confirm** to complete the acknowledgement.

8.5 Acknowledge multiple alarms

Procedure

Step 1. Use the sorting and filtering controls to display the necessary set of alarms in the list.

Step 2. Select all of the alarms you want to acknowledge by checking the box at the left edge of the list. Click the check box again to unselect any alarm.

Note: To select all alarms in the list, check the check box in the title row. Click the check box again to unselect all alarms in the list.

Note: The number of alarms you have selected, as well as the total number of alarms, is indicated at the lower right of the Alarms page.

Step 3. At the upper right of the Alarms page, click the **Table settings & actions** button.

Step 4. Select **Acknowledge** from the list.

Step 5. Click **Confirm** to complete the acknowledgement for all alarms.

8.6 Deleting a single alarm

Prerequisites

An alarm cannot be deleted unless it has first been cleared.

Procedure

Step 1. Find the alarm in the list using the sorting and filtering controls.

Step 2. At the right side of the row, click the **Table row actions** button.

Step 3. Select **Delete** from the list.

Note: The **Delete** option is not displayed for an alarm that has not been cleared.

Step 4. Click **Confirm** to complete the acknowledgement.

8.7 Deleting multiple alarms

Prerequisites

An alarm cannot be deleted unless it has first been cleared.

Procedure

Step 1. Use the sorting and filtering controls to display the necessary set of alarms in the list.

Step 2. Select all of the alarms you want to delete by checking the box at the left edge of the list. Click the check box again to unselect any alarm.

Note: To select all alarms in the list, check the check box in the title row. Click the check box again to unselect all alarms in the list.

Note: The number of alarms you have selected, as well as the total number of alarms, is indicated at the lower right of the Alarms page.

Step 3. At the upper right of the Alarms page, click the **Table settings & actions** button.

Step 4. Select **Delete** from the list.

Step 5. Click **Confirm** to complete the acknowledgement for all alarms.

Note: If some of the alarms you selected were not eligible for deletion, only those that were eligible are deleted by this operation. Ineligible alarms are not deleted. No error message displays in this case.

8.8 Suppress a single alarm

Procedure

Step 1. Find the alarm in the list using the sorting and filtering controls.

Step 2. At the right side of the row, click the **Table row actions** button.

Step 3. Select **Suppress** from the list.

Step 4. Click **Confirm** to complete the acknowledgement.

Note: By default, suppressed alarms are not displayed in the alarms list. Unless you have selected to show all alarms, suppressing an alarm causes it to vanish from the alarms list.

8.9 Suppressing multiple alarms

Procedure

Step 1. Use the sorting and filtering controls to display the necessary set of alarms in the list.

Step 2. Select all of the alarms you want to delete by checking the box at the left edge of the list. Click the check box again to unselect any alarm.

Note: To select all alarms in the list, check the check box in the title row. Click the check box again to unselect all alarms in the list.

Note: The number of alarms you have selected, as well as the total number of alarms, is indicated at the lower right of the Alarms page.

Step 3. At the upper right of the Alarms page, click the **Table settings & actions** button.

Step 4. Select **Suppress** from the list.

Step 5. Click **Confirm** to complete the acknowledgement for all alarms.

Note: By default, suppressed alarms are not displayed in the alarms list. Unless you have selected to show all alarms, suppressing alarms causes them to vanish from the alarms list.

8.10 Viewing alarm history

Procedure

Step 1. Find the alarm in the list using the sorting and filtering controls.

Step 2. At the right side of the row, click the **Table row actions** button.

Step 3. Select **History** from the list.

Expected outcome

EDA opens the Alarm History window, which shows all events pertaining to the selected alarm including the following details:

- Cleared (yes/no)
- Last change date/time
- Probable cause
- Remedial action

You can use the standard sorting and filtering controls to manage the list.

9 EDA query language (EQL)

EDA supports queries using a syntax that is collectively referred to as the EDA Query Language, or EQL.

In EDA, a query consists of:

- a Table that identifies the overall set of data being queried.

  Note: The Table is the only mandatory element of any query.

- a Selector that defines a set of fields to return (along with any functions to run on those fields).
- a Filter that restricts the set of results to return.
- a Sort that indicates the order in which results should be returned.
- a Limit that restricts the number of results to return.
- a Frequency that indicates the minimum period after which to automatically update the query results.

For example:

- .namespace.alarms.current-alarm
- .namespace.alarms.current-alarm where (severity = "critical")
- .namespace.alarms.current-alarm where (severity = "critical") order by [type]
- .namespace.alarms.current-alarm where (severity = "critical") limit 5
- .namespace.alarms.current-alarm where (severity = "critical") order by [type] sample milliseconds 500

EDA also supports queries using Natural Query Language.

Queries and namespaces

Query results are always constrained to the set of namespaces to which the current user has access permissions. By default a system administrator can see query results spanning all namespaces; but users with fewer namespace privileges see results from only some namespaces, or only a single namespace.

If you have permission to access multiple namespaces, you can cite one or more specific namespaces as part of the filter to constrain the result to those namespaces.

Related topics

Namespaces

9.1 Elements of a query

A query using EQL can include the following elements.

Table

A Table is specified in jspath notation, with a `Table` boundary at all lists and containers within a `Topo Node` schema, or within containers/lists provided by `StateEngine` scripts or external gRPC publishers via `StateController`.

In simple terms, each node within the jspath is its own table: `.node` is a table, `.node.srl` is a table, and `.node.srl.interface` is a table.

A `Table` can be identified in one of the following formats:

- a complete jspath without keys. For example: `.namespace.node.srl.interface.subinterface`.
- a jspath with keys. For example: `.namespace.node{.name=="dut1"}.srl.interface{.name== "ethernet-1/1"}.subinterface{.index=="0"}`.
- a mix of the two. For example: `.namespace.node.srl.interface{.name=="ethernet-1/ 1"}.subinterface`.

Note: `.node.srl` is only relevant for SR Linux devices. SR OS devices publish to `.node.sros`, and `StateEngine` apps that normalize data should publish to `.node.normal`.

In cases where keys are omitted, a wildcard key is assumed. This means the last example could be rewritten as:

- `.namespace.node{.name=="*"}.srl.interface{.name=="ethernet-1/ 1"}.subinterface{.index=="*"}`

This could also be written as:

- `.namespace.node.srl.interface.subinterface where ..name="ethernet-1/1"`

Note: A key value that is an integer is assumed to be a string, unless some other schema information is available (that is, from a data model).

Selector

A Selector is denoted by the `fields` keyword, where the value is an array of fields to return, along with any functions to run.

- These fields must exist in the `Table` that is being queried, or the query fails.
- For example `.namespace.node.srl.interface FIELDS [admin-state, description] ORDER BY [oper-state ascending natural]`.
- No fields other than those defined are returned. If no fields are selected then all fields from the table are returned.
- The `fields` keyword must precede any `where` or `order by` keywords.

A set of functions can assist with evaluation and aggregation. For example:

- `isSet()` to evaluate if a field is set.
- `average()` to evaluate the average of a field matching a `Filter` over time (the time window here is currently fixed to the current set of data).
- `count()` to return the count of unique combinations matching a `Filter`.
- `sum()` to sum the values for a field matching a `Filter`.
- `max()` to return the maximum seen value for a field matching a `Filter`.

Filter

A Filter is a string defining any filters to use. A Filter is defined with a `where` term. A few rules are used:

- A `Filter` consists of an ordered set of fields, operators, values, and keywords.
  - Keywords may be capitalized or may not For example, both `and` and `AND` are valid.
  - Operators include =, `!=`, <=, >=, >, <, ~, `is not`, `is`, `in`, `not in`, / (division), + (addition), - (subtraction), * (multiplication), `%` (modulo), ^ (power/exponent).
    - `is` and `is not` evaluate similarly to = and `!=`, except that they allow the `EMPTY` keyword, . For example: `is not EMPTY`.
    - `in` is provided an array of values, for example `.node.srl.interface where (oper-state in ["up", "down"])`.
- Field names in a `Filter` are unquoted, and values are quoted where they are strings, and unquoted when they are integers
  - For example, `.namespace.node.srl.interface where (oper-state = "up")`.
  - For example, `.namespace.node.srl.interface where (ifindex = 49150)`.
- A Filter may string together multiple criteria through the use of `()`, and the keywords `AND`, and `OR`.

  Note: Note that even when using a single where statement it must be contained within `()`.

  - For example: `.table where ((oper-state = "down" and mtu = 1500) or oper-state = "up")`.
- A `Filter` can query ancestor keys and values by referencing their full jspath.
  - For example, to query the field `oper-state` in a direct parent, the query could read `.namespace.node.srl.interface.subinterface where (../oper-state = "up")`.
  - A Filter could also query a grandparent with `../../`, a great grandparent with `../../../`, and so on.
  - For example, to add `Filter` criteria for a parent key, the full jspath can be used: `.namespace.node.srl.interface.subinterface where (.node.name = "dut1")`.
- A Filter can query any child or parent containers, but cannot navigate into lists. For example, to query the field `error-count` in the `statistics` container of a parent, the query could read `.namespace.node.srl.interface.subinterface where (../statistics.error-count > 0)`.
- Evaluations against an unset field yield `True` for not equal and `False` for equal.
  - For example, assuming `.namespace.node.srl.interface.oper-state` (with valid keys) would exist with the current value up.
    - This query (again with valid keys) would return `True` and return the interface: `.node.srl.interface where (oper-state = "up")`.
    - This query (again with valid keys) would return `False` and not return the interface: `.node.srl.interface where (oper-state != "up")`.
  - Taking the same example, where `oper-state` was not set:

- This query (again with valid keys) would return `False` and not return the interface: `.namespace.node.srl.interface where (oper-state = "up")`.
- This query (again with valid keys) would return `True` and return the interface: `.namespace.node.srl.interface where (oper-state != "up")`.

Sort

A Sort is similar to a Filter, but instead of describing how to select data, it describes how to return data. A Sort is denoted by the `ORDER BY` keywords which controls the ordering (sorting) of data.

- A Query may include a single `ORDER BY` keyword, where the value is an array of (fields, sorting algorithms, and directions) which are evaluated in the order they are presented.
  - For example `.namespace.node.srl.interface ORDER by [oper-state ascending natural]`.
  - The second value may be either `ascending` or `descending` (or short forms `asc`, `desc`).
  - The third value is optional and may be either `natural`.

Limit

A Limit restricts the number of results that are returned. It is denoted by the `limit` keyword. A Limit is processed after any other operations (perhaps most relevant, the Sort operation).

- A `limit` accepts a single integer value.
- This can be combined with Sort to get the 'top' N results, or the 'bottom' N results, where N is the value provided to the `limit` keyword.
- The maximum value for `limit` is 1000, and the minimum value is 1. Any values above or below this return an error.

Frequency

A Frequency allows you to control the rate at which data is returned, and is denoted by the `delta` keyword.

- The `delta` keyword must be passed two values - one denoting the units used, and another the actual value.
- For example, `.namespace.node.srl.interface.traffic-rate where (in-bps != 0) delta seconds 1` means "do not update the client more than once every 1 second."
- The value is the minimum period at which results are updated for the query.
- Valid units are `seconds` and `milliseconds`.

9.2 Natural-language queries

When creating a query in EDA, you also have the option of writing the query in natural language. With a natural-language query, you can ask questions of EDA such as:

- List all up interfaces
- List all interfaces that have an MTU of 9232, sorted by interface name

- What statistics are available on interfaces
- Show me any interfaces with error counters above 0
- Show me the unique reasons interfaces are down
- Show me the unique reasons interfaces are down, and count the unique values
- Show me all of my processes sorted by memory usage descending
- Show me the total numbers of packets sent on all interfaces
- Show me the number of MAC addresses on subinterfaces on "leaf-1-1", include the interface name

Note: Currently, natural-language queries are resolved only against the `.node.srl` table.

9.3 Creating a query with EQL

About this task

The page in the EDA UI on which to create queries is located at **Main** → **Queries**.

Procedure

Step 1. From the **Query** drop-down, select **EQL Query**.

Step 2. Enter an expression using EDA Query Language (EQL), as described in Elements of a query.

- Begin the query with a period (.).
- As you begin typing the query, EDA offers suggestions for the next element in the expression.
- The finished query must specify a Table in jspath notation. This table identifies the overall set of data being queried. Optionally, the query can also include:
  - a Selector that defines a set of fields to return (along with any functions to run on said fields).
  - a Filter that restricts the set of fields to return.
  - a Sort that indicates the order in which data should be returned.
  - a Limit that restricts the number of results to return.
  - a Frequency that indicates the minimum period after which to automatically update the query results.

Step 3. When you have completed the query expression, click **Query** to view the results.

Note: Results are limited to the first 1,000 matches.

9.4 Creating a query with natural language

About this task

The page in the EDA UI on which to create queries is located at **Main** → **Queries**.

Procedure

Step 1. From the **Query** drop-down, select Natural Language Query.

Step 2. Type your question using simple language (not necessarily English). Your question must specify something to return information about (such as nodes, links, or other network objects).

> **Note:** Currently, natural-language queries are resolved only against the `.node.srl` table.

Optionally, your question can also specify:

- conditions those objects must meet.
- ways to sort the returned data.
- a limit on how many results to return.
- a time period after which to automatically update the query results.

Step 3. When you have finished typing your query, click **Query** to view the results.

> **Note:** EDA renders your natural language question in EDA Query Language, and displays the EQL expression immediately below the query field.

> **Note:** Results are limited to the first 1,000 matches.

10 Transactions

In EDA, a transaction is a set of changes that must succeed or fail together. If one item in a transaction fails, the whole transaction is deemed to have failed and any successful changes within the transaction are rolled back to their previous state.

Any resource configurations you create, change, or delete in EDA must be committed in order for those changes to be applied to the participating resources.

- Any single configuration change can be committed immediately on its own, in which case it constitutes a transaction consisting of just that change.
- However, you do not need to commit each configurations individually as you create it. EDA allows you add any new, changed, or removed configuration to a growing set of configuration changes that you want to commit together as one collective transaction. When you are ready you can then commit the whole transaction, applying the entire set of configuration changes together.
  The complete transaction, including all of its constituent configuration changes, then succeeds or fails as a unit. If any part of the transaction fails, the whole transaction is rolled back. You can then resolve the blocking issue, and apply the transaction again.

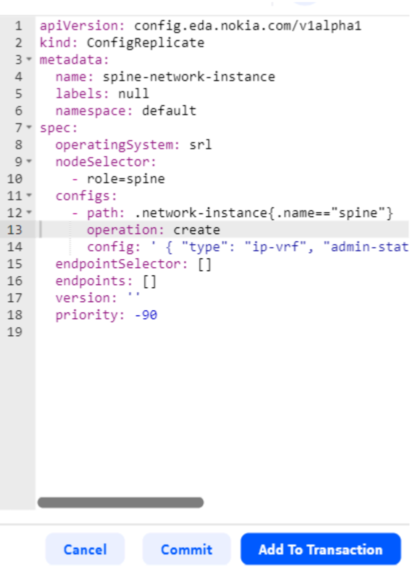
Within EDA, ConfigEngine is the main service behind transactions. Its job is to compute the complete set of resources that must be modified, deleted, or created as part of the transaction; ensure all dependencies and outputs are captured; and then transact the updates, thereby generating changes for NPPs and other controllers.

After a transaction has been successfully processed it is written to EDA's Git repo for persistence, becoming the new accepted state of the infrastructure.

Commit options for a typical resource

When you configure a resource in EDA, you are always presented with the following options to either commit the change, or add it to a transaction:

Figure 23: A sample resource ready to commit



```
apiVersion: config.eda.nokia.com/v1alpha1
kind: ConfigReplicate
metadata:
  name: spine-network-instance
  labels: null
  namespace: default
spec:
  operatingSystem: srl
  nodeSelector:
    - role=spine
  configs:
    - path: .network-instance{.name=="spine"}
      operation: create
      config: ' { "type": "ip-vrf", "admin-stat
  endpointSelector: []
  endpoints: []
  version: ''
  priority: -90
```

- **Commit**: clicking this button immediately commits the change to EDA to be applied to the target device.
- **Add to Transaction**: clicking this button adds the current resource configuration to the current transaction. It is not committed, or configured on the target device, until the transaction itself is committed.

10.1 Transactions drop-down panel

The **Basket** icon is displayed at the top of every page in the EDA GUI. It indicates when any transactions are pending, and can be used to open the Transaction drop-down.

When one or more transactions are pending (containing at least one configuration and ready to commit), the **Basket** icon is highlighted and displays a count of pending transactions.

Figure 24: The basket showing one pending transaction

You can open the **Transactions** drop-down panel by clicking the **Basket** icon.

Thee **Transactions** drop-down panel gives you a fast way to:

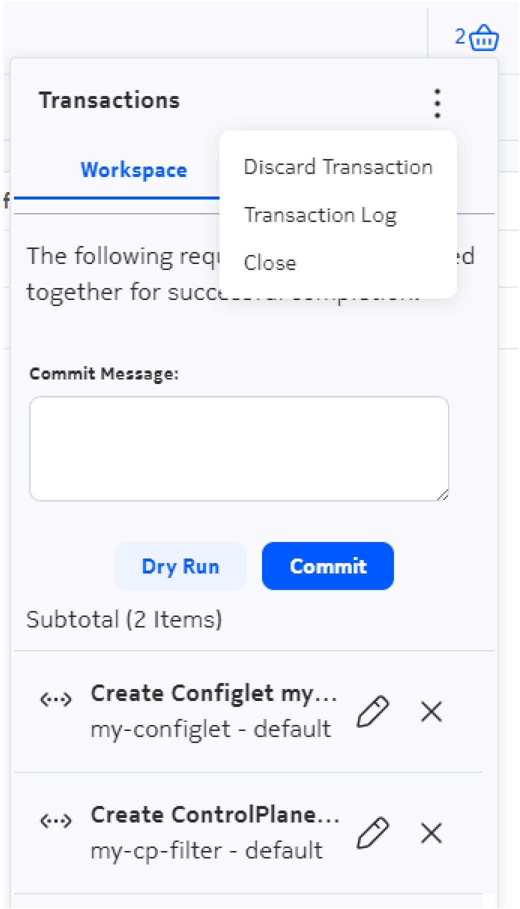
- review the contents of any pending transactions
- perform a dry run of any pending transaction
- add a commit message to a pending transaction
- commit any pending transaction
- discard any pending transaction
- view a list of recently completed transactions

You can also use this drop-down panel to manage individual configurations within the transaction. You can:

- edit a single configuration
- remove a configuration from the transaction
- restore a removed configuration back into the transaction

You also use this drop-down panel to view the EDA Transaction Log.

Figure 25: The transaction drop-down panel



Dry runs

In EDA, you can perform a Dry Run of any pending transaction.

In a Dry Run, the system does not send any of the configuration changes to the managed nodes. However, it executes the transaction against EDA's stored information about each participating node, and validates the transaction against that data. This can reveal anticipated configuration issues if the transaction were to proceed normally, and allow you to troubleshoot any errors before committing the transaction on actual nodes.

Commit

Clicking **Commit** commits the entire transaction. The complete transaction, including all of its constituent configuration changes, then succeed or fail as a unit.

If any part of the transaction fails, the whole transaction is rolled back. You can then resolve the blocking issue, and apply the transaction again.

Recent

Click on the **Recent** tab to view a list of the most recent transactions. A more complete list of transactions is available from the Transactions log.

Configuration actions

The following actions are available for each configuration within the transaction:

- **Edit** the configuration: Click the **Edit** icon to open the original resource configuration page so that you can modify the configuration details.
- **Remove** the configuration: Click the **Minus** icon to remove only this configuration from the overall transaction.
- **Restore** the configuration: after removing a configuration from the transaction, click the **Plus** icon to restore the configuration as part of the overall transaction.

Additional actions

From the Actions menu of the Transactions drop-down, you can perform any of the following:

- **Discard Transaction**: discards the transaction and all configurations it included. The transaction still appears in the transaction log, but its Success status is False.
- **Transaction log**: select this action to open the Transactions page, showing a list of recent transactions.

10.2 Transactions page

The **Transaction** page displays the content of the Transactions log, providing an overview of transactions performed in the EDA system. From this page you can:

- View the list of transactions
- View a detailed summary of individual transactions
- View the precise configuration changes ("Diffs") in JSON format that were included in any transaction
- revert a successful transaction
- restore a reverted transaction

The Transactions page does not display pending transactions; a transaction must be completed before it appears in this list.

Transactions list

The Transactions list displays all of the transactions recorded in the Transactions log.

The Transactions log does not include the entire EDA transaction history. It is limited to:

- 128 successful transactions
- 1024 failed transactions total (this number is larger than successful transactions because failed transactions are often more important for debugging reasons)
- 25 transactions per user

The following information displayed in the transactions list:

Table 23: Transaction information

Column	Description
Display ID	An identifier for the transaction, in the form "Transaction <number>".
Description	An optional description provided at the time the transaction is committed.
Completion Timestamp	The date and time the transaction was completed.
Success	Whether the transaction was successful, or failed.
State	The state of the completed transaction.
Dry Run	Whether a Dry Run has been performed for this transaction. In a Dry Run, the system does not send any of the configuration changes to the managed nodes. However, it executes the transaction against EDA's stored information about each participating node, and validates the transaction against that data.
Commit Hash	A unique string that identifies this string in the EDA repository.

For each row in the Transaction list, a set of actions available from the **Table row actions** menu:

- **Summary**, which opens the EDA Transaction Summary page showing detailed information for one transaction.
- **Revert**, which creates a new transaction that effectively undoes the selected transaction. The result is to set the resources affected by the commits in this transaction to their state before the transaction took place.
- **Restore**, a powerful command that restores all resources to their state at the completion of the selected transaction.

Transactions summary

The Transaction Summary page shows detailed information about a single transaction. This includes:

- The completion status, the name of the transaction, any description provided, and a timestamp for its completion.
- An Error Summary showing the number of
  - General errors
  - Intent errors
  - Node configuration errors
- A Custom Resource (CR) Summary, showing the number of:
  - Input CRs (new resource configurations enacted as part of the transaction)
  - Changed CRs (existing resources that were reconfigured as part of the transaction)
  - Intents Run
  - Nodes with Changes

Transaction details

The Errors and CRs that are summarized at the top of the Summary page can be viewed in detail in the Transaction Details panel. Different tabs display lists of input CRs, changed CRs, intents run, nodes with changes, and errors.

Transaction diffs

From the breadcrumb at the top of the Transactions page you select Diffs to view the precise details of the configurations changed as part of this transaction.

Each resource added or changed as part of this transaction is displayed, along with the JSON code representing the new or changed lines in the respective configuration data.

10.3 Adding a resource configuration to a transaction

Procedure

Step 1. Create any resource in the EDA GUI.

Step 2. Click **Add to Transaction** at the bottom of the configuration page.

Expected outcome

The configuration for this resource is added as an item in the current transaction. It is not committed until you commit the entire transaction.

Related topics

Committing a transaction

10.4 Committing a transaction

Procedure

Step 1. Click the **Basket** icon at the top of any page in the EDA GUI.

> **Note:** When at least one transaction is pending, the basket icon is highlighted and displays a count of the pending transactions available to commit.

Step 2. Optionally, click **Dry Run** to perform a dry run of this transaction.

> **Note:** In a Dry Run, the system does not send any of the configuration changes to the managed nodes. However, it executes the transaction against EDA's stored information about each participating node, and validates the transaction against that data.

Step 3. Optionally, edit any resource configuration that is part of the current transaction by clicking the **Edit** icon to the right of the transaction. This opens the original resource configuration page so that you can modify the configuration details.

Step 4. Optionally, remove any resource configurations that are currently part of the transaction that you do not want to commit at this time. Remove a transaction by clicking the **Minus** icon to the right of the transaction.

Step 5. Click **Commit**.

Expected outcome

EDA commits all of the resource configurations included in the transaction. If any configuration fails, the transaction is halted and rolled back. See the **Transactions** page for details about transaction success, configurations and nodes affected and any errors that may have occurred.

11 Topology

A common visualization used to describe relationships between structures is a graph or topology diagram. A topology consists of a set of nodes, links, and endpoints, with one or more toggle-able overlays and badges to indicate various types of status for the displayed objects.

A node in the topology is an anchor for endpoints, and the relationship between endpoints is described using links.

In EDA, a topology can represent the relationships between any set of resources that have a structured relationship. The most common form of this is a physical topology, which EDA uses to illustrate how managed `TopoNode` resources interact with each other using `TopoLink` resources.

The most common topology visualization, and the one currently supported by EDA, is a tree in which:

- Nodes and links are included in the tree
- Each level of the tree is denoted as a "tier"
- A tier may have one or more groups of resources
- Relationships between tiers are drawn based on links present

Nodes

In a physical topology, a node is simply a termination point for endpoints. For example, in a data center a topology can be represented using:

- A leaf switch that is abstracted using a `TopoNode`, which becomes a "node" in the topology.
- The interface connecting to a spine is abstracted using an `Interface`, which becomes an "endpoint" in the topology.
- The physical cable plugged into the interface is abstracted using a `TopoLink`, which becomes the "link" in the topology.

Within the EDA GUI's topology display, nodes can display badges and adopt a particular shading based on a selected overlay.

Links

Any links with endpoints on nodes selected is drawn as links connecting those nodes in the topology illustration. Other links may extend into an abstract "edge" icon.

Endpoints

An endpoint is one end of a link, and is commonly used in physical topologies.

11.1 The Topologies page

You can view the topology page by selecting **Topologies** in EDA's main menu.

Figure 26: The Topologies page

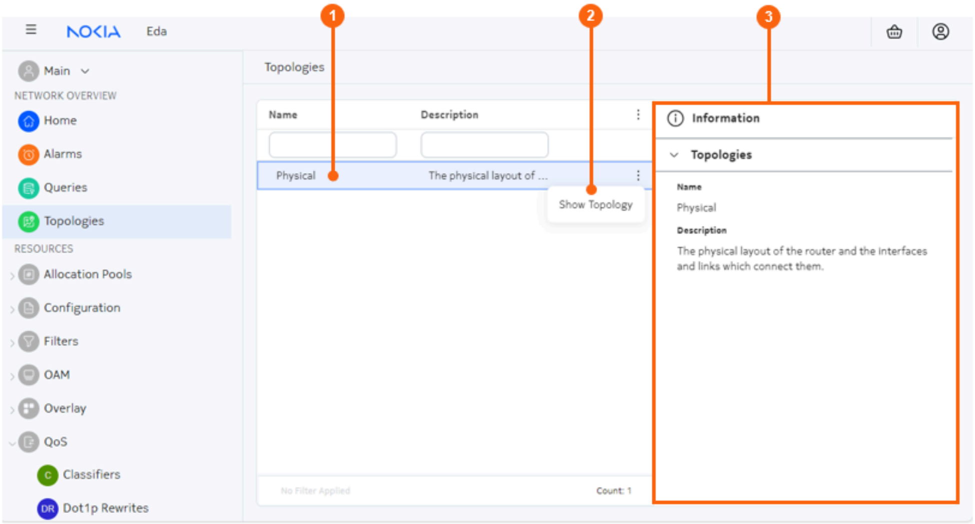


Table 24: Elements of the Topologies page

#	Name	Function
1	Topology list	Select an available topology. Currently, the Physical topology is supported.
2	Table row action list	Select to take an action for the current row. Currently the **Show Topology** action is supported.
3	Information panel	Like most page of the EDA GUI, the Topologies list includes an information panel that displays details about the currently selected object.

The topology illustration

When you select **Show Topology** from the actions on the Topologies page, EDA displays a graphical representation of the topology including nodes and links.

Figure 27: The topology illustration

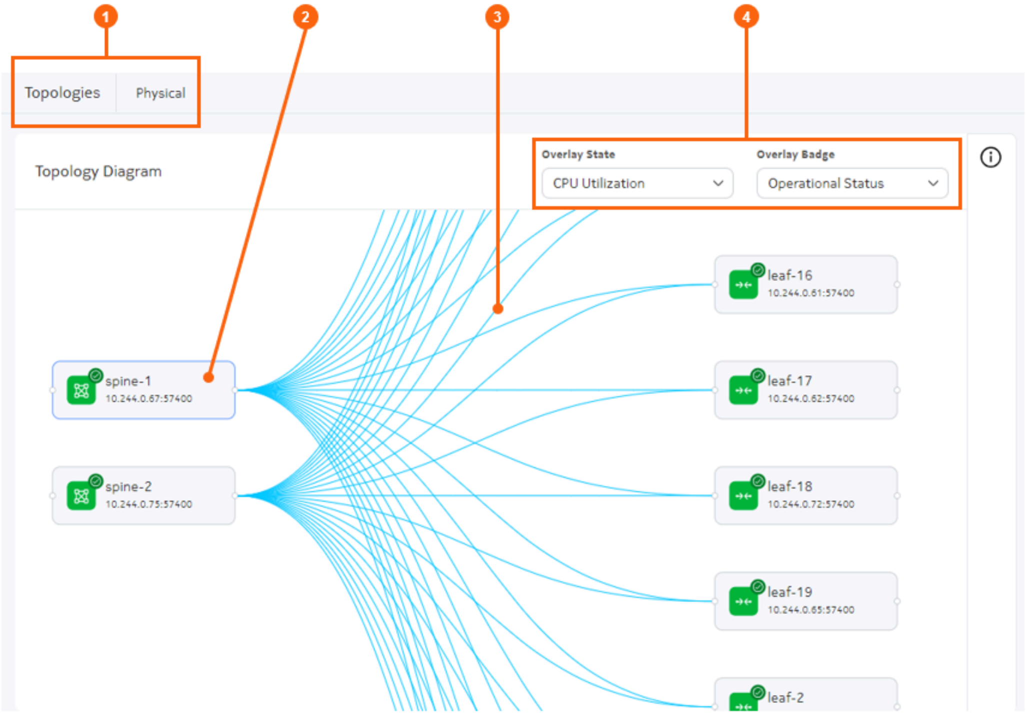


Table 25: Elements of the topology illustration

#	Name	Function
1	Topology breadcrumb	Displays the name of the topology currently being displayed. You can click **Topologies** to return to the Topologies page.
2	Node	This is an example of a node within the topology. Click on any node to view details about that node in the **Information** panel.
3	Links	These are examples of links within the topology. Click on any link to view details about that link, the connected nodes, and the link endpoints in the **Information** panel.
4	Overlay and Badge selectors	Use this drop-down list to select an available overlay and / or set of badges. Each overlay applies shading to the topology illustration to indicate the related status of all nodes and links. Badges display on the nodes with in the topology, and the badge icon and color indicate the related status of each node.

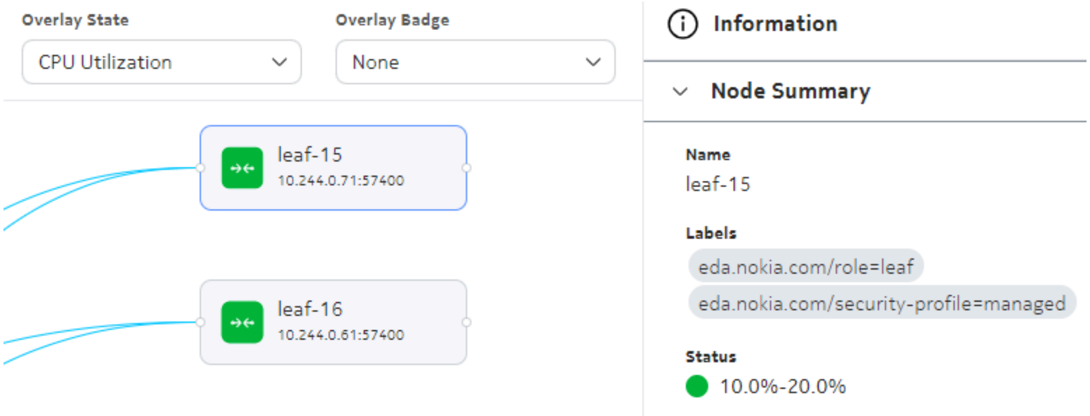
Overlays

Overlays are specific to a topology, although the same functional overlay may exist for different topologies. For example, EDA's status overlay shows the operational state of the resources within the topology, and shades them a corresponding color in the topology illustration.

In EDA, you can toggle overlays on and off. When available, multiple overlays can be enabled concurrently.

More details about the significance of overlay shading is available in the **Information** panel for each node.

Figure 28: CPU Utilization Overlay and information panel



Overlay State: CPU Utilization

Overlay Badge: None

Information

Node Summary

Name: leaf-15

Labels: eda.nokia.com/role=leaf, eda.nokia.com/security-profile=managed

Status: ● 10.0%-20.0%

The following overlays are available for physical topologies in EDA:

- The Status overlay indicates the operational status of each TopoNode and TopoLink in the topology. This status derived from the status of NPP connectivity to the node, and the operational state of underlying interfaces making up the TopoLink. As with other overlays, updates are constantly updated to reflect moment-by-moment status.
- The CPU utilization overlay shades nodes based on the percentage of CPU utilization.
- The Deviations overlay indicates whether the current, detected configuration of a node or link differs in some way from the configuration expected by EDA. This can happen when methods external to EDA are used to configure resources.

More details about the significance of the overlay shading for a selected object is available in the **Information** panel for a badged node.

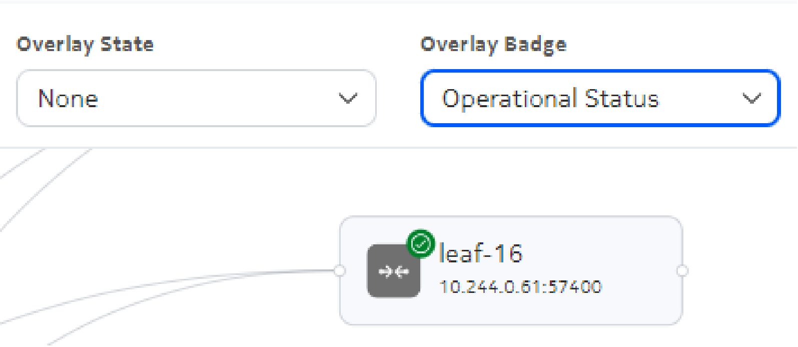
Badges

EDA applications may provide badges for display on the nodes with the topology display. Each badge consists of an icon or a single-digit character (to identify the information type) and a set of colors (to indicate status related to that information type). Where multiple badges are available and enabled, they display in a series at the upper right of the node image.

More details about the significance of a badge is available in the **Information** panel for a badged node.

For some badges, you can also hover your mouse over the badge to see more information about its meaning.

Figure 29: A badge indicating that a node's Operational Status is 'Synced'



Overlay State
None

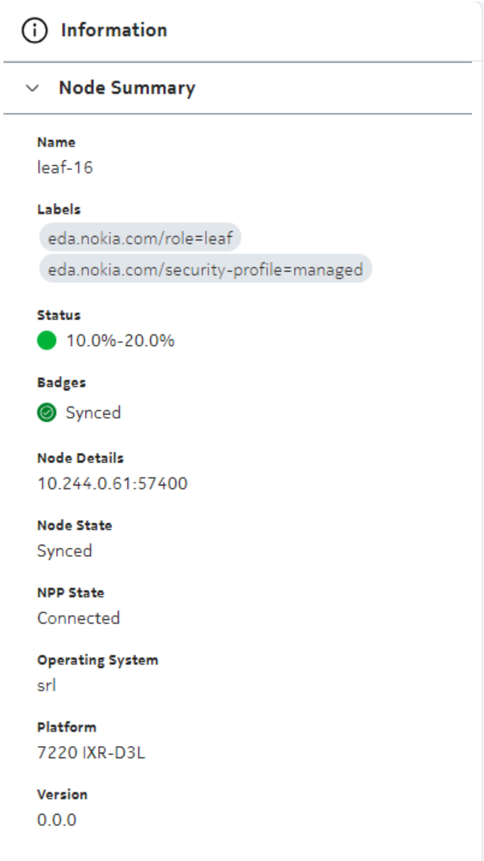
Overlay Badge
Operational Status

Information panel

As with many pages in the EDA GUI, an information panel is available on the right side of the topology illustration. Expand this panel to view detailed information about a selected object within the illustration.

This can be a useful way to see the specific meaning of overlays shading or badges within the topology illustration.

Figure 30: A sample information panel display



Information

Node Summary

Name
leaf-16

Labels
eda.nokia.com/role=leaf
eda.nokia.com/security-profile=managed

Status
● 10.0%-20.0%

Badges
✓ Synced

Node Details
10.244.0.61:57400

Node State
Synced

NPP State
Connected

Operating System
srl

Platform
7220 IXR-D3L

Version
0.0.0

11.2 Viewing a topology

About this task

The page in the EDA UI in which to view network topologies is located at **Main** → **Topologies**. This page displays a list of available topologies.

Procedure

Step 1. Select a topology from the list.

Step 2. Click the **Table row actions** icon at the right edge of the list and select **Show Topology** from the actions list.

Expected outcome

The Topology Diagram displays, showing a graphical representation of the nodes and links within network topology.

Step 3. Optionally, select an overlay by clicking the **Overlay State** drop-down and selecting an overlay from the list.

Step 4. Optionally, select a badge type by clicking the **Overlay Badge** drop-down and selecting a badge type from the list.

Step 5. To view details about an individual node, select the node and then expand the **Information** panel.

Note: When the **Information** panel is open, you can click other objects to see their information in the panel.

Step 6. To view information about a link within the topology, click the link. If it is not already expanded, expand the **Information** panel.

The panel shows information about:

- the nodes at either end of the link, and their status
- the inter-switch links that connect the nodes, and their status
- the endpoints of each link and their status.

Note: For any displayed item in the link **Information** panel, click the **More** icon to open the configuration window for that link. From there you can see complete details about the link configuration.

12 EDA applications

Applications (apps) can be simple micro-Python apps with one or more CRDs exposed through the different engines, or they can be more complex like the Connect or VMware plugin apps that deploy entire operators and controllers as pods in Kubernetes. In general, all intents are considered apps.

12.1 App Store

The EDA App Store is used to manage the apps inside EDA. An app is an extension that can be deployed inside an EDA cluster to extend the functionality of EDA.

In the EDA UI page, you can access the **EDA App Store** page from **System Administration** → **Store**.

The **App Store** page provides the following views for apps:

- **All Packages**: This view displays all the app packages known to the App Store. This view is the default when you open the **App Store** page.
- **My packages**: This view shows all the installed apps in the current EDA deployment. Select this view by clicking the **All Packages** drop-down list and selecting **My packages**.
- app page view: This view provides information about an app. Click any app from the **All Packages** or **My packages** views to open the page for a specific app.
  This view displays the following information about an application:
  - Description
  - Version
  - Author
  - Publish
  - Supported operation system
  - Category
  - The latest version
  - The installed version, if the app is installed
  - The application source

12.1.1 Apps

In the EDA UI, you can interact with apps from the **System Administration** → **AppStore** page.

Apps are defined by a manifest that contains the following information about the app:

- Name or title
- Vendor or author
- Version

- Published Date (build date of the App of that version)
- Source code link
- Git references and paths
- License link
- Short description
- Main readme link
- Documentation link
- Screenshot links
- Artifacts
- Containers
- Scripts
- CRs
- CRDs
- UI design/configuration
- Dependencies/Requirements

12.1.1.1 Installing an application

About this task

After the EDA software is installed, the App Store has access to the catalogs and registries. All apps available for installation display in the App Store.
In the EDA UI, perform this task from the **System Administration** → **Store** page.

Procedure

Select the app that you want to install and click **Install**.

The App Store creates an `AppInstall` custom resources in the EDA environment. The App Store takes this resource and takes the appropriate actions to read the information from the app's manifest and ensure that the appropriate data is available to EDA, which can include the code of the app, custom resource definitions and more.

You can monitor the progress of the installation on the UI or the API. If at any time the installation fails, the previous steps are rolled back where possible.

12.1.1.2 Uninstalling an app

About this task

In the EDA UI, perform this task from the **System Administration** → **Store** page.

Procedure

Step 1. Select the app that you want to uninstall and click it.

Step 2. Click **Uninstall package**.

Expected outcome

The App Store uninstalls the app.

12.2 App management

The App Store relies on the following resources to manage apps:

Catalog A catalog is a git repository that contains the manifests of apps.

Registry An App Store Registry is a container registry that contains OCI compliant images of an app. It is where the app content and code are uploaded as a single OCI image.

App installer An app installer is a workflow resource that informs the App Store it must install an app in the EDA environment.

12.2.1 Catalogs

A catalog is a git repository that contains the manifests of applications. A manifest contains all the details of an app. The App Store builds a list of all available apps using the manifests of all the catalogs registered in EDA.

In the EDA UI, you can interact with catalogs from the **System Administration** → **Catalogs** page.

The **Catalogs** page displays all the catalogs available in EDA. You can double-click a catalog to view its details or edit it. For more information about how to manage a catalog, see Managing a catalog.

12.2.1.1 Creating a catalog credentials secret

About this task

The credentials created in this procedure is used in Adding a catalog to the App Store.

Procedure

To create a catalog credentials secret, update the following `Secret` resource file:

```
apiVersion: v1
kind: Secret
type: Opaque
metadata:
  name: your-creds # A unique secret name
data:
  username: <base64(username)> # Base64 encoded username
  password: <base64(password or token)> # Base64 encoded password/token
```

- Provide a unique name in the `name` field.
- Provide base64 encoded values for the `data` fields.

12.2.1.2 Adding a catalog to the App Store

Prerequisites

You must have already created the authentication secret as described in Creating a catalog credentials secret.

About this task

In the EDA UI, you can add a catalog from the **System Administration** → **Catalogs** page.

Procedure

Step 1. Click **Create**.

Step 2. Configure metadata for this resource.
Set the following fields:

- **Name**
- **Labels**
- **Annotations**

Step 3. Configure the specifications for this catalog.
Set the following parameters:

- **Authentication Secret Reference**: Provide the Kubernetes secret that contains the credentials to connect to the Catalog git repository over HTTPS.
- **Description**
- **Remote Type**: select from the drop-down list
- **Remote URL**: provide the path to the catalog, the URL of the git repo where the catalog resides
- **Skip TLS Verify**: by default, TLS verification is enabled; set this parameter to skip TLS verification
- **Title**: provide the name of the catalog as you want it to appear in the **Catalogs** page

Step 4. Click **Add to Transaction**.

12.2.1.3 Managing a catalog

About this task

In the EDA UI, you can manage a catalog from the **System Administration** → **Catalogs** page.

Procedure

Locate the a catalog and at the end of its row, click the Table Row actions menu to select the action that you want to take.
You can take the following actions:

- View app details
- Edit details about the catalog
- Delete the catalog

Before deleting a catalog, ensure that no apps are using it.

12.2.2 Registries

The actual code and resources of an app are stored in an OCI-compliant image. This image is stored in a container registry. This registry must be known to the EDA deployment so the app store can pull the image and use the data in the image to deploy the app. This information is provided to EDA in the form of a registry custom resource.

In the EDA UI, you interact with registries from the **System Administration** → **Registries** page. The **Registries** page displays all available registries in the EDA system.

12.2.2.1 Creating the registry credentials secret

About this task

Before you can add a registry to the App Store, you must create a Kubernetes secret that contains the credentials to connect to the registry over HTTPS.

Procedure

Update the `Secret` YAML files.

- Provide a unique name in the `name` field.
- Provide base64 encoded values for the `data` fields.

```
apiVersion: v1
kind: Secret
type: Opaque
metadata:
  name: your-creds # A unique secret name
data:
  username: <base64(username)> # Base64 encoded username
  password: <base64(password or token)> # Base64 encoded password/token
```

12.2.2.2 Adding a registry

About this task

In the EDA UI, you can add a registry from the **System Administration** → **Registries** page.

Procedure

Step 1. Click **Create**.

Step 2. Configure metadata for this resource.
Set the following fields:

- **Name**
- **Labels**
- **Annotations**

Step 3. Configure specifications for this registry.

- **Authentication Secret Reference**: specify how to authenticate with the remote registry. This setting is a reference to a Kubernetes secret, which contains a username and password as data.
- **Mirror**: the URL for mirror for this registry
- **Remote URL**: provide the FQDN or IP address for the registry. This setting should only contain the FQDN or IP address of the registry, not a URL path.
- **Skip TLS Verify**: enable this parameter to skip TLS verification
- **Title**: provide the name of the registry as you want it to appear in the **Registries** page

12.2.2.3 Managing a registry

About this task

In the EDA UI, perform this procedure from the **System Administration** → **Registries** page.

Procedure

Locate the registry and at the end of its row, click the Table Row actions menu to select the action that you want to take. You can take the following actions:

- View details about a registry
- Edit details about the registry
- Delete the registry
  Before deleting a registry, ensure that no apps are using it.
- Duplicate the registry

13 Security

This section describes EDA security features, including how to secure access to EDA and its resources, node security, EDA infrastructure security, and certificate management.

13.1 Securing access to EDA

EDA uses Keycloak, a well-known and secure solution, for its identity and access management. Authentication is required to interact with EDA.

EDA implements authorization through role-based access control (RBAC) includes the following elements:

Users

Individuals with access to the system. Each user has a user information profile to store information about them. System administrators can assign users to user groups.

User groups

A collection of users organized according to the type of activities they are meant to perform. You assign resource access rights to user groups through user roles. When you assign a role to a user group, all access rights defined in the role are inherited by the users of the group.

Roles

Specifies which resources users or associated user group members can access. You assign network resource access to roles through resource groups. Each member of a group can perform the roles specified for that group.
A role that exists in a namespace is referred to as a *role*. A role that exists cluster wide (that is, it is not in a namespace) is referred to as a *cluster role*.

A user can belong to more than one group, and a group can be assigned multiple roles.

13.1.1 Roles

A role specifies which network resources users or associated user group members can access.

In EDA, a `Role` resource lives within a namespace, while a `ClusterRole` resource applies cluster-wide and spans namespaces.

Note: While similar in concept, EDA roles and cluster roles are not the same roles and cluster roles in Kubernetes.

`Role` and `ClusterRole` resources are created by a system administrator and are referenced in groups that exist in Keycloak (or a remote directory), which are in turn associated with users. As with Kubernetes, the a cluster role spans namespaces, while a role lives within a namespace.

A role controls access to EDA resources by defining one or more match rules and corresponding action to take when there is a match.

Related topics

Rules

13.1.1.1 Rules

EDA supports the following types of rules for `Role` and `Cluster Role` resources:

- Resource rules: control access to EDA API resources.
- Table rules: control access to the database tables.

  Note: Write access is not supported on a table rule.

- URL rules: controls access to API endpoints that are not specific to resources or tables.

Resource rules

Resource rules define access to EDA and Kubernetes resources exposed via the API server. These rules are relevant for resource-aware API endpoints including:

- `/openapi/v3/apps/..`
- `/core/transaction/v1`
- `/apps/..`

A resource rule is defined by the following parameters in the `Role` or `ClusterRole` resource:

- **API groups**: Identifies the EDA API groups for the resources controlled by the rule, in the format `apigroup/version`. An asterisk `*` indicates match any API group.
- **Permissions**: Specifies the permissions for the EDA resources specified by the rule. Set to **none**, **read**, or **readwrite**.
- **Resources**: The resource names of the resources controlled by the rule.
  An asterisk `*` indicates wildcard, which means match any resource in the specified API group.

Table rules

Table rules are similar to resource rules, except that they are relevant to the API endpoints used for querying the EDA database. A table rule is defined by the following parameters:

- **Path**: Specifies the path to the database table for which this rule applies.
  The `/` character at the end of the path indicates the final portion of the URL path can be anything, if the prefix matches.

  `*//` at the end of the path indicates that the URL path can be anything if the prefix matches.
- **Permissions**: Specifies the permissions for the EDA resources specified by the rule. Set to **none** or **read**; writing to the database tables is not allowed.

URL rules

URL rules define generic enforcement of URL paths exposed by an API server. A URL rule is defined by the following parameters:

- **Path**: the API server-proxied URL path to which this rule applies.
  The `/` character at the end of the path indicates the final portion of the URL path can be anything, if the prefix matches.

  `*//` at the end of the path indicates that the URL path can be anything if the prefix matches.

A `path` may contain a single asterisk `*` to include fields (but not children) of the path. For example, `/core/admin/*` includes all fields available directly under `admin`, but would not include child paths like `/core/admin/groups/{uuid}`.

- **Permissions**: Specifies the permissions for the API server-proxied URL path for the rule. Set to **none**, **read**, or **readwrite**

Rule behavior

EDA rules are additive. Users are granted the combined permission of all rules in the roles assigned to their user groups. If a request does not match any rule, it is implicitly denied.

'None' permission rules acts as an override; these are enforced before any other rule.

EDA supports the principles of additive permissions and least permission, (aligning with Kubernetes recommendation described in https://kubernetes.io/docs/concepts/security/rbac-good-practices/#least-privilege.

The following are best-practice recommendations for administrators:

- Create explicit read/readWrite rules for the resource, table, and URLs rule required for a role.
- Avoid wildcard read/readWrite rules where possible. A wildcard gives access to resources that exist today and resources that may exist in the future.
- Avoid ‘None’ rules where possible. If resource, table, or URL access is not required for a role, do not include a matching rule for that resource/table/URL (that is, implicit deny). 'None' rules are explicit denials. They have priority over all permissive rules assigned to the user, including the rules defined in other groups and roles.

13.1.1.2 Creating a cluster role

About this task

A `ClusterRole` resource defines a set of permissions to access EDA resources. In the EDA UI, you can create a cluster role from the **System Administration** → **Cluster Roles** page.

Procedure

Step 1. Click **Create**.

Step 2. Configure metadata for this resource.

Set the following fields:

- **Name**
- **Labels**
- **Annotations**

Step 3. Provide an optional description for this cluster role.

Step 4. In the **Resource Rules** section, click **+ Add**. Create resource rules for this cluster group.

a. Under **API Groups**, click **Add Item** to create the list of API groups.

b. Under **Permissions**, from the drop-down list, select **None**, **read**, or **readWrite**.

For related information, see Roles.

c. Under **Resources**, click **+ Add** to specify the resources on which this rule applies.

You can enter the * character, which means match any resource in the matching API groups.

d. Click **Save**.

Step 5. In the **Table Rules** section, click **+ Add**.

a. Provide the path to the database table to which this rule applies.

b. Under **Permissions**, from the drop-down list, select **None** or **read**.

c. Click **Save**.

Step 6. In the **URL Rules** section, click **+ Add**. Create resource rules for this cluster group.

a. Provide the path to the API server proxied URL to which this rule applies.

b. Under **Permissions**, select **None**, **read**, or **readWrite** from the drop-down list.

c. Click **Save**.

Example: `ClusterRole` resource

```
kind: ClusterRole
metadata:
  name: basic
  labels: {}
  annotations: {}
spec:
  description: ''
  resourceRules:
    - apiGroups:
        - core.eda.nokia.com/v1
      permissions: read
      resources:
        - '*'
    - apiGroups:
        - fabrics.eda.nokia.com/v1alpha1
      resources:
        - fabrics
      permissions: readWrite
    - apiGroups:
        - fabrics.eda.nokia.com/v1alpha1
      resources:
        - '*'
      permissions: read
  urlRules:
    - path: /core/transaction/v1/**
      permissions: read
  tableRules:
    - path: .namespace.node.**
      permissions: read
```

13.1.1.3 Default cluster role

EDA provides a default `system-administrator` cluster role with the following configuration:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    kubernetes.io/bootstrapping: rbac-defaults
  name: system-administrator
```

```
rules:
- apiGroups:
  - "*"
  resources:
  - "*"
  permissions: readWrite
urlRules:
- path: "/**"
  permissions: readWrite
tableRules:
- table: ".**"
  permissions: read
```

13.1.1.4 Creating a role

About this task

The `Role` resource defines a set of permissions to access EDA resources. The **Role** resource exists within a namespace. In the EDA UI, you can create a cluster role from the **System Administration** → **Roles** page.

Procedure

Step 1. Click **Create**.

Step 2. Configure metadata for this resource.

Set the following fields:

- **Name**
- **Labels**
- **Annotations**

Step 3. Provide an optional description for this cluster role.

Step 4. In the **Resource Rules** section, click **+ Add**. Create resource rules for this cluster group.

a. Under **API Groups**, click **Add Item** to create the list of API groups.

b. Under **Permissions**, from the drop-down list, select **None**, **read**, or **readWrite**.

For related information, see Roles.

c. Under **Resources**, click **+ Add** to specify the resources on which this rule applies.

You can enter the `*` character, which means match any resource in the matching API groups.

d. Click **Save**.

Step 5. In the **Table Rules** section, click **+ Add**.

a. Provide the path to the database table to which this rule applies.

b. Under **Permissions**, from the drop-down list, select **None** or **read**.

c. Click **Save**.

Step 6. In the **URL Rules** section, click **+ Add**. Create resource rules for this cluster group.

a. Provide the path to the API server proxied URL to which this rule applies.

b. Under **Permissions**, select **None**, **read**, or **readWrite** from the drop-down list.

c. Click **Save**.

Example: `Role` resource

```
apiVersion: core.eda.nokia.com/v1
kind: Role
metadata:
  annotations: {}
  name: ns-admin
  namespace: eda
  labels: {}
spec:
  resourceRules:
    - apiGroups:
        - '*'
      permissions: readWrite
      resources:
        - '*'
  tableRules:
    - path: .**
      permissions: read
  urlRules:
    - path: /**
      permissions: readWrite
  description: ''
status: {}
```

13.1.2 User groups

A user group associates multiple users with a role, enabling them to access EDA resources. An admin user can create user groups and assign a specific role to each group according to the type of network activities the user group is meant to perform. When a role is assigned to a user group, all users within the group have the same access to resources, as specified by the role.

EDA comes with a default user group called system-administrator. Users who belong to this group can:

- Create, update, and delete local groups.
- Assign local users to local groups.
- Assign remote users to local groups.
- View all users and their group memberships.

13.1.2.1 Creating a user group

About this task

In the EDA UI, perform this task from the **System Administration** → **User Management** page. From the **Users** drop-down list, select **User Groups**.

Procedure

Step 1. Click **Create**.

Step 2. Set the **Name** parameter with the name of the user group.

Step 3. In the **Users** section, click **Add** to add users to the user group.

Step 4. From the **Assigned Roles** drop-down list, select a roles to assign to the user group.

You can only select one role.

Step 5. Click **Save**.

13.1.3 Users

Users are individuals with access to the EDA system. Users gain access to application and network resources through the user groups to which they are assigned. A user can be assigned to more than one user groups, either locally or through an external directory.

Individual users can also be assigned roles directly, without membership to a user group.

Types of users

The following types of users interact with EDA components:

- Local users are created locally on the EDA system and authenticated using Keycloak.
- Remote users are configured on a remote directory, such as a Lightweight Directory Access Protocol (LDAP) server, that the system queries to authenticate remote users when they try to log in. For more information, see Remote directories
- Node users are configured access to a set of topnodes. are deployed to a set of TopoNodes. A NodeUser resource configures a node user's password, SSH keys, and group bindings.

Default admin user

EDA comes with a default local user called admin. The admin user is assigned to the system-administrator group and can perform the following functions:

- create, update, and delete users (except for the admin user)
- manually set a password for users during creation
- modify the password of the admin user, and perform other functions other than modifying its group
- disable or enable non-admin users without deleting the

13.1.3.1 Creating a new local user

About this task

In the EDA UI, perform this task from the **System Administration** → **Users and Groups** page.

Procedure

Step 1. Click **Create**.

Step 2. In the **User Information** section, enter the required information the new user.

- a username to identify the user
- the user's first and last name
- the user's email address

Step 3. Click **Set Password**.

In the dialog box that opens, provide a password and confirm it. By default, the password is temporary and a user must log in and provide a new password for the newly created account.

Step 4. Assign this user to one or more user groups.

From the **Assigned User Groups** drop-down list, select an existing user group. Optionally, you can create a user without assigning the user to a user group. Later, you can add the user to a user group.

Step 5. Click **Save**.

13.1.3.2 Managing user accounts

About this task

In the EDA UI, perform this task from the **System Administration** → **Users and Groups** → **Users** page.

Note: A user with system-administrator privileges cannot delete the built-in admin user or modify its groups or roles.

Procedure

Locate the user and at the end of its row, click the Table Row actions menu to select the action that you want to take.

- Select **Edit** to update details for a user such as user first name and last name, email, password, and assigned user groups. You can also enable or disable a user.
- Select **Set Password** to set a new password.

13.1.3.3 Changing your password

About this task

Perform this task from any page on EDA UI.

Procedure

Step 1. Click the user icon at the upper right of the screen and select **Change Password**.

Step 2. When prompted, log in again with your credentials.

Step 3. Enter your new password and confirm it.

Step 4. Click **Save**.

13.1.4 Password policies

The system enforces a default system-wide user password policy for local users. The password policy does not apply to users authenticated from remote directories.

The default password policy includes password aging rules, password complexity rules, password history, and user lockout rules. An admin user can update the default policy settings as needed. The default policy also applies to the admin user.

Note: Nokia recommends that system administrators configure a password policy for production deployments.

13.1.4.1 Modifying the default password policy

About this task

A user who is assigned the system-administrator role can modify the default password policy from the **System Administration** → **RBAC** → **Password Policy** page in the UI.

Procedure

Step 1. Click **Create**.

Step 2. Modify any of the following fields:

- **Minimum Length**: the minimum length of a password
- **Minimum Lowercase Characters**: the minimum number of lowercase characters
- **Minimum Special Characters**: the minimum number of symbols or special characters
- **Password History** the number of passwords to keep and validate against
- **Minimum Uppercase Characters**: the minimum number of uppercase characters
- **Minimum numbers**: the minimum number of numerical characters
- **Allow Username** : specifies whether the username can be used as a password
- **Password Expiry**: the duration, in days, for a password to remain valid
- **Hashing Algorithm**: select from **ARGON2**, **PBKDF2-SHA512**, **PBKDF2-SHA256**, or **PBKDF2** (the default)

Step 3. Modify the lockout policy settings.

- **Maximum Login Failures**: the maximum consecutive failed login attempts before account lockout
- **Failure Wait Duration**: duration, in seconds, to wait after reaching the maximum login failures before retry is allowed
- **Permanent Lockout**: Lock the account permanently after maximum number of failed logins
- **Reset time (Required)**: Duration, in seconds, after which failed login attempts are reset

13.1.5 Remote directories

EDA supports the use of external directories that the system can use to authenticate users who were not created locally on the system.

Note: Currently, EDA supports the configuration of remote directories using the EDA API; the UI does not support the configuration of remote directories.

EDA supports:

- the configuration of up to five directories

- multiple directory types - KeyCloak supports Lightweight Directory Access Protocol (LDAP) and Active Directory
- the ability to indicate whether group assignment can be derived from directory and provide any directory-relevant configuration to assist with this mapping
- the ability to specify whether the directory is synchronized or queried every time. Additional configuration of the synchronization interval.
- read-only access, no support for write-back to directories (for password changes, for example).

When a remote directory is configured, system administrators can continue to create local users in EDA, but these newly created users are not pushed to the remote server. Directories are ordered and this order can be changed, for example, specifying that the system does local authentication first, and then query a directory if local authentication does not match.

EDA supports the configuration of the following attributes for a remote server:

- Type (LDAP)
- TLS (LDAPS)
- Bind credential
- Bind DN
- Timeout
- URL
- User search filter
- User object classes
- Username attribute
- User DN
- ID/UUID attribute
- Vendor
- Enabled
- Import
- Pagination
- Scope

13.2 Platform security

This section describes the security features for the EDA platform and infrastructure.

13.2.1 Unique Keycloak client secret per installation

To avoid the risk of a secret revealed at one customer can affect the installations of other installations, internal secrets used by the different EDA components must unique for each installation. This practice is especially important for the Keycloak secrets that used by the API server to configure and communicate with the Keycloak API server.

13.2.2 Changing the Keycloak secret

About this task

By default, a unique secret is generated per installation. Use this procedure to regenerate a new Keycloak secret.

Procedure

Step 1. From your web browser, navigate to `{EDA_URL}/core/httpproxy/v1/keycloak`.

Step 2. Log in with the Keycloak administrator username and password.

Step 3. From the **Keycloak** drop-down list on the upper left, select **Event Driven Automation eda**.

Step 4. Select **Clients** from the menu on the left.

Step 5. Select "eda" in the client table in the main web page area.

Step 6. Select **"Credentials"** in the tab bar containing, **"Settings/Keys/Credentials/Roles/..."**

Step 7. Note the current "Client Secret".

Step 8. Click **Regenerate** to generate a new random value for the secret.

13.2.3 Changing the Keycloak admin password

About this task

Use this procedure to change the Keycloak admin password.

Procedure

Step 1. From your web browser, navigate to `{EDA_URL}/core/httpproxy/v1/keycloak`.

Step 2. Log in with the current Keycloak administrator username and password.

Step 3. From the user drop-down list on the upper right, select **Manage Account**.

Step 4. From the menu on the left, select **Account Security** → **Signing In**.

Step 5. Click **Update** next to **My Password**.

Step 6. Configure a new password and save it.

Step 7. Generate the Base 64 hash of the new password.

Step 8. Using a system with access to the Kubernetes API of the EDA deployment, execute the following command:

```
kubectl -n eda-system patch secret keycloak-admin-secret -p
        '{"data": { "password": "<NEW BASE64 HASH>" }}'
```

Step 9. Restart the Keycloak service.

Example

```
kubectl -n eda-system rollout restart deployment/eda-keycloak
```

13.3 EDA certificate management

EDA plays the following roles in certificate management:

- Generates, signs, and distribute certificates, keys, and trust-bundles to EDA components and managed nodes
- Generates a rootCA if no sub CA/CA provided during installation
- Generates a self-signed certificate for API/AS if no certificate is provided during installation; create a self-signed issuer to bootstrap a CA issuer
- Generates a certificate signing request (CSR) and sign certificates for nodes
- Generates a CSR and sign certificates for all EDA services

13.3.1 Trust bundles

Trust bundles are collections of root certificates that a client or a server trusts. EDA uses root certificates to sign generated certificates before distributing them to applications or nodes. EDA applications use these root certificates to validate the authenticity of the certificates presented by a Transport Layer Security (TLS) peer during a TLS handshake.

Trust bundles are auto-generated during installation. EDA uses the following trust bundle (CertManager) issuers:

- internal issuer
- API issuer
- node issuer

Trust bundles are distributed to EDA components using the `CertManager Bundle` CR. The `Bundle` CR allows a user to create a trust bundle from multiple sources (ConfigMaps, Secrets) and make them available to an application through a different ConfigMap than the sources.

During installation, applications that need to use trust bundles can mount the resulting ConfigMap to have access to the assembled trust bundle.

13.3.1.1 Internal issuer

The internal issuer is a CertManager certificate authority (CA) issuer that is responsible for signing the key pairs used by EDA pods for internal communication, including both client and server interactions.

Example: Internal Issuer

The internal issuer includes the CertManager `Certificate` and `Issuer` CRs.

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: eda-internal-ca
spec:
  isCA: true
  commonName: eda-internal-ca
  subject:
    organizations:
      - Nokia
    organizationalUnits:
      - NI
  secretName: eda-internal-ca
  secretTemplate:
    labels:
      eda.nokia.com/ca: "internal"
  usages:
    - digital signature
    - cert sign
    - key encipherment
    - server auth
    - client auth
  privateKey:
    algorithm: ECDSA
    size: 256
  issuerRef:
    name: eda-root-ca-issuer
    kind: Issuer
    group: cert-manager.io
---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: eda-internal-issuer
spec:
  ca:
    secretName: eda-internal-ca
```

13.3.1.2 API issuer

The API issuer is the CertManager issuer that signs the key pairs used by EDA API pods for exposing a TLS HTTP server.

This issuer is configured using the `CertManager Certificate` and `Issuer` CRs.

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: eda-api-ca
spec:
  isCA: true
  commonName: eda-api-ca
  subject:
    organizations:
      - Nokia
    organizationalUnits:
      - NI
  secretName: eda-api-ca
  secretTemplate:
    labels:
      eda.nokia.com/ca: "api"
  usages:
    - digital signature
    - cert sign
    - key encipherment
    - server auth
    - client auth
  privateKey:
    algorithm: ECDSA
    size: 256
  issuerRef:
    name: eda-root-ca-issuer
    kind: Issuer
    group: cert-manager.io
---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: eda-api-issuer
spec:
  ca:
    secretName: eda-api-ca
```

User-provided key pair

During installation, you can optionally provide a key pair (public and private keys) for the API server to use as certificate and key. In this case, you are responsible for the rotation of the provided key-pair.

To rotate the public and private keys, create a Kubernetes `Secret` resource called `eda-api-user-certs` that contains the certificate and key under `tls.crt` and `tls.key`, respectively. For example:

```
apiVersion: v1
kind: Secret
metadata:
  name: eda-api-user-certs
  labels:
    eda.nokia.com/ca: api
type: kubernetes.io/tls
data:
  # base 64 encoded certificate
  tls.crt: |
    LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUNVakNDQWJzQ0FnMytNQTBHQ1NxR1NJYjNE
    UUVCQlFVQU1JR2JNUXN3Q1FZRFZRUUdFd0pLVURFT01Bd0cKQTFVRUNCTUZWRzlyZVc4eEVEQU9C
    Z05WQkFjVEIwTm9kVzh0YTNVeEVUQVBCZ05WQkFvVENFWnlZVzVyVkVSRQpNUmd3RmdZRFZRUUxF
    dzlYWldKRFpYSjBJRk4xY0hCdmNuUXhHREFXQmdOVkJBTVREMFp5WVc1ck5FUkVJRmRsCllpQkRR
    VEVqTUNFR0NTcUdTSWIzRFFFSkFSWVVjM1Z3Y0c5eWRFQm1jbUZ1YXpSa1pDNWpiMjB3SGhjTk1U
    TXcKTVRFeE1EUTFNVE01V2hjTk1UZ3dNVEV3TURRMU1UTTVXakJMTVFzd0NRWURWUVFHRXdKS1VE
    RVBNQTBHQTFVRQpDQXdHWEZSdmEzbHZNUkV3RHdZRFZRUUtEQWhHY21GdWF6UkVSREVZTUJZR0Ex
    VUVBd3dQZDNkM0xtVjRZVzF3CmJHVXVZMjl0TUlHYU1BMEdDU3FHU0liM0RRRUJBUVVBQTRHSUFE
    Q0JoQUo5WThFaUhmeHhNL25PbjJTbkkxWHgKRHdPdEJEVDFKRjBReTliMVlKanV2YjdjaTEwZjVN
    Vm1UQllqMUZTVWZNOU1vejJDVVFZdW4yRFljV29IcFA4ZQpqSG1BUFVrNVd5cDJRN1ArMjh1bklI
    QkphVGZlQ09PekZSUFY2MEdTWWUzNmFScG04L3dVVm16eGFLOGtCOWVaCmhPN3F1TjdtSWQxL2pW
    cTNKODhDQXdFQUFUQU5CZ2txaGtpRzl3MEJBUVVGQUFPQmdRQU1meTQzeE15OHh3QTUKVjF2T2NS
    OEtyNWNaSXdtbFhCUU8xeFEzazlxSGtyNFlUY1JxTVQ5WjVKTm1rWHYxK2VSaGcwTi9WMW5NUTRZ
    RgpnWXcxbnlESnBnOTduZUV4VzQyeXVlMFlHSDYyV1hYUUhyOVNVREgrRlowVnQvRGZsdklVTWRj
    UUFEZjM4aU9zCjlQbG1kb3YrcE0vNCs5a1h5aDhSUEkzZXZ6OS9NQT09Ci0tLS0tRU5EIENFUlRJ
    RklDQVRFLS0tLS0K
  # base64 encoded private key
  tls.key: |
    RXhhbXBsZSBkYXRhIGZvciB0aGUgVExTIGNydCBmaWVsZA==
```

13.3.1.3 Node issuer

The node issuer is a CertManager issuer that is responsible for signing the key pairs that EDA installs on the nodes to secure the configured gRPC servers. This issuer is configured using the CertManager `Certificate` and `Issuer` CRs, as shown in the following example:

```
apiVersion: cert-manager.io/v1
kind: Certificate
```

```
metadata:
  name: eda-node-ca
spec:
  isCA: true
  commonName: eda-node-ca
  subject:
    organizations:
      - Nokia
    organizationalUnits:
      - NI
  secretName: eda-node-ca
  secretTemplate:
    labels:
      eda.nokia.com/ca: "node"
  usages:
    - digital signature
    - cert sign
    - key encipherment
    - server auth
    - client auth
  privateKey:
    algorithm: ECDSA
    size: 256
  issuerRef:
    name: eda-root-ca-issuer
    kind: Issuer
    group: cert-manager.io
---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: eda-node-issuer
spec:
  ca:
    secretName: eda-node-ca
```

During installation, a provider can supply the rootCA (public and private keys) that EDA uses as an issuer for the nodes key-pairs. The user does this by creating a secret and an CA issuer that references the secret.

13.3.2 Certificate key pairs

A certificate key pair consist of a `Certificate` and `Key` objects. Certificate key pairs are used by either servers or clients within an application.

A certificate key pair is generated and distributed for the following uses:

- for EDA components (client or server) to use for internal communication
- to install on a node or to rotate a certificate on a node
- for an EDA API server

13.3.2.1 Certificate key pairs for EDA components

For EDA components, Cert-Manager (an x.509 certificate controller) generates, signs, and distributes the signed certificates and keys to the relevant pods.

EDA uses Cert-Manager injects the generated certificate and key into a volume mounted to the pod where the application is running. Using this driver ensures that the private key and corresponding signed certificate is unique to each pod and is stored on disk to the node on which that pod is scheduled. This driver also handles renewal of live certificates as needed.

The life cycle of the certificate key pair matches that of the pod; the certificate is issued when the pod is created and destroyed when the pod is terminated.

Example

The example below shows a pod CR with two sets of certificate and key pairs that requests the signing of each of the certificates from Cert-Manager. The CSI driver generates a private key, request a certificate from Cert-Manager based on the `volume` attributes.

```
apiVersion: v1
kind: Pod
metadata:
  name: eda-internal-sample-app
  labels:
    app: eda-internal-sample-app
spec:
  containers:
    - name: eda-internal-sample-app
      image:
      volumeMounts:
      - mountPath: "/var/run/eda/tls/external"
        name: tls-external
      - mountPath: "/var/run/eda/tls/internal"
        name: tls-internal
  volumes:
    - name: tls-external
      csi:
        driver: csi.cert-manager.io
        volumeAttributes:
          csi.cert-manager.io/issuer-name: eda-external-ca
          csi.cert-manager.io/dns-names: ${POD_NAME}.${POD_NAMESPACE}.svc.cluster.local
    - name: tls-internal
      csi:
        driver: csi.cert-manager.io
        volumeAttributes:
          csi.cert-manager.io/issuer-name: eda-internal-ca
          csi.cert-manager.io/dns-names: ${POD_NAME}.${POD_NAMESPACE}.svc.cluster.local
```

13.3.2.2 Certificate key pairs for nodes

EDA uses the gNSI Certz or gNOI certificate management protocols to generate, distribute, and rotate the certificate and key and cert-manager to sign the certificate for nodes.

The bootstrap server uses the following parameters for rotating certificates (these settings cannot be modified in the current release):

- `RotationThreshold`: the percentage of remaining certificate validity at which the bootstrap server rotates the certificate. This value is set to 50%.
- `CriticalFailedRotationThreshold`: if the certificate rotation fails, this is the percentage of remaining certificate validity at which the bootstrap server generates a critical alarm. This value is set to 70%.
- `BackoffTimer`: the backoff duration (set to 60 seconds) to wait between failed rotation attempts.

Nodes that support gNSI Certz

During bootstrap, the initial configuration provided to the node must contain at least two gRPC servers to handle the gNMI and gNSI services:

- `bootstrap` server: the gNMI server configured with the network-instance `mgmt` and port `50052`; it uses the default TLS profile (`default-tls-profile: true`)
- `mgmt` server: the gNSI server configured with network-instance `mgmt`, port `57400`; it uses a dummy TLS profile named `EDA`

Initial certificate key pair generation

The bootstrap server adds the initial certificate key pair to the node using the gNSI protocol. The bootstrap server discovers the node by periodically sending gNMI capabilityRequest messages on port 50052 with the flag `skip-verify: true`. When the node is discovered, the bootstrap server creates the certificate key pair and rotates the TLS profile called `EDA` on the node.

Certificate key pair rotation

The bootstrap server rotates the node certificate and key pair when the certificate is about to expire, according to the configured validity time and the rotation threshold.

The bootstrap server monitors the certificate expiry in the `TLSProfile` CR and trigger its renewal.

Change in node issuer certificate triggers certificate rotation

When the node issuer certificate changes, the bootstrap server checks if the node issuer certificate stored in the certStore (security git repo) is different from the one in the issuer secret. If the certificate is different, the bootstrap server triggers the node certificate rotation regardless of the certificate validity time.

EDA alarms on certificate rotation failure

EDA generates alarms on it detects failure in the rotation of certificates. It generates an initial (Major) alarm when the certificate rotation fails at the rotation threshold and a second (Critical) alarm when it `Critical FailedRotationThreshold`.

These alarms should be cleared when the node certificate is successfully rotated. The alarm should specify the node name, the profile name that failed to be rotated.

13.4 Privacy considerations

- From a privacy perspective, EDA stores user information securely in a database. This information includes the username, password, email, first name, last name, and login times of each user in the system. Additionally, user activity is logged securely for security and support perspectives. The information is not processed or shared outside of the deployed EDA environment. A backup contains the same information for the purpose of restoring the users if a restore is required.

  Note: Ensure that you store the backup information securely and limit access to both the running environment and any backup storage environment.

- Handle all environments containing privacy sensitive information according to the regulations that apply to the location and users of the system and the data.

14 Administration

System administration encompasses a range of activities that configure or manage the EDA system. Some of these operations are performed within the EDA user interface; others are performed as command-line operations on the server that hosts the application.

14.1 Image management

The `Image` resource is the workflow that is used to change image of the operating system on nodes, via an upgrade or downgrade. For more information about workflows, see Workflows and workflow definitions.

Note: The Image workflow in currently supported only using EDA APIs; the UI does not currently support re-imaging.

You can use the `Image` workflow to perform the following tasks:

- Reimage a single node, a list of nodes by name, a set of nodes using a label selector, or tranches of nodes, including support for canaries
- Perform a configurable set of pre and post checks to verify:
  - that all interfaces are up (with an option to use a label selector
  - that all default BGP peers are up
  - reachability on ISLs
  - reachability between system addresses

In the `Image` resource, provide the input for the following fields:

- `Type`: the type of workflow. Set to `node`, `nodeselector`, or `tranche`
- `nodeProfile`: set to the destination `NodeProfile` resource.
- `version`: set to the destination version in the `NodeProfile`.

The following settings are optional:

- `prompt`: specifies when the workflow prompts an operator to continue. Set to `AfterPreChecks` or `AfterPostChecks`. These options can force a prompt even when checks pass.
- `checks`, a container that includes options for pre and post checks:
  - `skip`: indicates that checks should be skipped (not run).
  - `force`: indicates that checks should be run without prompts, even if checks fail.
  - `checks`: lists the checks to run. If not provided, all checks are run. Valid checks are `Interface`, `DefaultBGP`, `PingISL`, and `PingSystem`.

Requirements for re-imaging nodes

Before running an image workflow, ensure that deviations in the system, if any, have all been accepted or rejected.

Ensure that node groups and node users have the necessary privileges to perform the operations. For example, the GNOI/GNSI privilege is required to execute operational commands such as reboot.

Re-imaging failure

If re-imaging fails, the workflow terminates with the appropriate reason. Operators are responsible for cleaning up any configuration created by the workflow, including the following recovery steps:

- Cleaning up drain policies created by the workflow.
- Removing any configlets created by the workflow.
- Reverting the node profile and setting the NPP mode if necessary.

Operators can attempt re-imaging by creating a new workflow.

14.1.1 Reimaging individual nodes

To reimage individual nodes using the `Image` workflow, provide the following input:

- `type`: set to `node`
- `nodes`: set to the name of the `TopoNodes` to be reimaged

Example: Workflow resource for re-imaging specific nodes

```
apiVersion: core.eda.nokia.com/v1
kind: Workflow
metadata:
  name: workflow-upgrade-leaf1
  namespace: eda
spec:
  type: operatingsystem-image
  input:
    type: node
    nodes:
    - mv-dc1-fabric1-leaf1
    nodeProfile: srlinux-24.10.1
    version: "24.10.1"
```

For related information, see Workflows and workflow definitions.

14.1.2 Reimaging nodes using labels

To reimage a set of nodes using a label selector, provide the following input:

- `type`: set to `nodeselector`.
- `nodeSelectors`: provide a list of label selectors to select `TopoNodes`.

Example: Workflow resource for re-imaging nodes using a label selector

```
apiVersion: core.eda.nokia.com/v1
kind: Workflow
metadata:
  name: workflow-upgrade-rack1
  namespace: eda
spec:
  type: operatingsystem-image
  input:
    type: nodeselector
    nodeSelector:
      - 'eda.nokia.com/redundancy-group=rack1'
    nodeProfile: srlinux-24.10.1
    version: "24.10.1"
    prompt:
    - AfterPreChecks
    drains:
      minimumWaitTime: 5
```

For related information, see Workflows and workflow definitions.

14.1.3 Reimaging node tranches

To reimage sets of groups of nodes with an ordered list of label selectors, provide the following input:

- `type`: set to `tranche`.
- `tranches`: set to a list of `nodeSelector`, nesting the node selector type
- `canaries`: optionally be used with tranches, specify as a pre-tranche to be reimaged before all others, and are subsequently the last to be undone in the event of a revert.

Imaging proceeds as follows:

1. Canaries are imaged first, executing any pre and post checks along with any waits. Assuming these operations succeed, the workflow continues.
2. Tranche with index 0 is imaged next, following the same run-to-completion workflow.
3. Tranche with the next index is imaged next, with this repeating until all tranches have been upgraded.

Example: Workflow resource for re-imaging tranches of nodes

```
apiVersion: core.eda.nokia.com/v1
kind: Workflow
metadata:
  name: workflow-upgrade-rack1-rack2
  namespace: eda
spec:
  type: operatingsystem-image
  input:
    type: tranche
    drains:
      skip: true
    tranches:
      - nodeSelector:
          - 'eda.nokia.com/redundancy-group=rack1'
        name: rack1
      - nodeSelector:
          - 'eda.nokia.com/redundancy-group=rack2'
        name: rack2
    nodeProfile: srlinux-24.10.1
    version: "24.10.1"
```

For related information, see Workflows and workflow definitions.

14.1.4 Node imaging checks

The `Image` workflow supports the following checks during node imaging:

- Verifying that interlink switch interfaces are operational. This check gets any `Interface` resource with the label `eda.nokia.com/role=interSwitch` where the current node is a member. The list of up interfaces is stored for comparison later.
- Verifying that BGP peers are up in the default network instance. As with interfaces, the list of up default BGP peers is stored for comparison later.
- Verifying connectivity on every ISL. This check triggers the ping workflow to run, passing in `isl` as the `pingType`.
- Verifying connectivity between all system addresses of nodes. This triggers the ping workflow to run, passing in `system` as the `pingType`.

These checks are executed before an upgrade batch takes place, and after the upgrade batch completes. If any check fails, the administrator is prompted to continue, but only after completing the execution of each test. If an operator rejects continuing in post checks, the image reverts to its previous version.

14.2 Technical support

On occasion, it may be necessary to troubleshoot technical issues that arise in EDA clusters. When working with Nokia technical support engineers, collaboration often requires sharing a set of background data about the EDA system that can help pinpoint and resolve the issue.

To help collect the necessary data, EDA includes a shell script you can use to automatically collect and package the necessary technical data about your system and the status of relevant components.

To execute the shell script:

- Open a shell to the `eda-toolbox Pod`
- Run `./tools/techsupport/techsupport.sh`

The output of the script is a gzipped tarball containing the following data:

- Logs from the cluster.
- Various information from Kubernetes, collected from all `Namespaces`:
  - all `Services` data, including YAML.
  - all `Pods` data, including YAML.
  - all `Nodes` data, including YAML. Note these are Kubernetes worker nodes, not EDA `TopoNodes`.
- Various information relating to EDA:
  - all CRDs, including YAML.
  - all resources of all CRDs, including YAML. This is essentially a full collection of all resources that are not native to Kubernetes.
  - all transactions, including YAML.
  - a system backup, which includes all git repositories.

14.3 Backup and restore

Critical systems are often backed up to ensure continuity in the event of an outage or other critical failure. Operators typically use backups for the following purposes:

- to replicate a customer issue when debugging
- to restore a cluster to a previous state (often via re-installation)

EDA's implementation for backup and restore include the following features:

- You can perform backups at any time with no outages or maintenance actions required.
- Backups are atomic and contain the lost known set of working configuration. The system waits until an in-progress transaction is completed before proceeding with a backup.
- You can restore into a dirty cluster and revert the cluster back to the state provided in the backup, auditing any resources as necessary.
- You can restore into a clean/freshly installed cluster.

Backups

In EDA, a backup is simply the copy of all git repositories in use at the time of the backup and an `Engine Config` resource file from the source that can be optionally restored.

The **edactl platform backup** command is used to create a backup. At a high-level, the this command does the following:

- Creates a tarball of all repositories, including the following:
  - Backup
  - Apps
  - User storage
  - Certificates
- Adds to this tarball the current `EngineConfig` resource file.
- Streams these files to the client performing the backup over gRPC.

The tarball created is created in the format `eda-backup-<cluster-member-name>-<date-and-time>.tgz` in the current directory. You can optionally provide a name for the tarball and provide an alternate destination by providing the name and the destination's path in the command.

Restore process

The `edactl platform restore` command restores a backup. When you initiate a restore operation, the ConfigEngine on the destination cluster performs the following tasks:

- Receives the restore request via gRPC. The request contains the complete tarball generated via a backup.
- Unpacks the backup, overwriting all content as it goes.
- Pushes backed up repositories and files to any server identified in `EngineConfig` resource.
- Restarts, relying on Kubernetes to restart.
- Starts again as if it had started clean from the repositories.

Note: A restore operation restarts the ConfigEngine, so use the command with caution.

14.3.1 Creating backups

About this task

You can create a backup using the `edactl platform backup` command.

Procedure

Step 1. From the eda-toolbox pod command line prompt, execute the **edactl platform backup** command.

Step 2. Copy the backup from the eda-toolbox pod to a safe location.

14.3.2 Restoring backups

Prerequisites

- The destination cluster must be running the same version as the cluster from where the backup was created.
- You must have rights and permissions on the cluster in which the backup is to be restored.

Procedure

Step 1. Copy the saved backup to the eda-toolbox pod.

Step 2. From the eda-toolbox pod, execute the **edactl platform restore <eda-backup.tar.gz>** command.

14.4 Redundancy

As part of critical infrastructure, EDA must be resilient in case of outages to continue to support the infrastructure. Outages can be caused by power outages, network outages, storage outages, or any other dependent infrastructure outages and EDA must be able to mitigate the loss of visibility and automation during these events. Outages can also impact the connectivity between members of an EDA cluster; in these cases, EDA needs to avoid split brain scenarios.

EDA provides resiliency via redundancy, using the following strategies:

- Localized restartability: assuming any application can fail at any time, and the system must reconcile. This approach is taken in general in EDA, and is quite relevant for services like ConfigEngine. In general, any service should be able to restart and the system converge back to a golden state. It is also true that on a failure of any EDA pod either Kubernetes or ConfigEngine should restart it.
- Localized redundancy and microservices: multiple instances of a common service with loadbalancing. This strategy limits localized outages, in most cases, only inflight requests are lost.
- Remote redundancy: multiple clusters (or cluster members depending on hierarchy). Typically referred to as geo-redundancy, where one or more cluster members are present and each one can operate the full load of management activities, with only one active at a time. In EDA, pushes to redundant sites are

not synchronous as long as changes are persisted in the majority of configured git servers. This does mean some inflight changes could be lost during a switchover.

Local redundancy

EDA supports automatic recovery of local services in the event of a failure. EDA leverages Kubernetes for deployment of its core services, which provides out-of-the-box redundancy when more than one worker node is available, with EDA services able to be scheduled or rescheduled to remaining available nodes during failures.

Cluster recovery

EDA supports cluster recovery by allowing the bootstrapping of a cluster from any member. This process removes all members, start the active member, and then add members back.

Remote redundancy

Remote redundancy is accomplished by configuring a set of members within the `EngineConfig` resource in `.spec.cluster.redundancy.members` context, and a credential to authenticate members at `.spec.cluster.redundancy.credential` context.

Synchronization occurs when changes are pushed to the set of git servers for backup.

Alarms

Support for the following alarms, generated only on the active cluster:

- Reachability to any member fails.Additional details should be included - ideally a user should be able to figure out if the issues relate to connectivity or authentication for example.
- Latency to a member is above a specified threshold.
- Any core-generated alarms from any standby member. These are forwarded to the active for the active to display, with the node set to the name of the member that raised it.

14.4.1 Geo-redundancy (remote redundancy)

EDA supports two concepts of remote redundancy that can be used together or separately:

Git redundancy

EDA supports remote redundancy through the backup of configuration information and data to a set of git servers and restoring backed up data from the same set of git servers.

The git servers are defined in the `.spec.git.servers` context of the `EngineConfig` CR. Whenever a change occurs in the system, the active ConfigEngine asynchronously pushes changes to all git servers, and from there, any other ConfigEngine can start with the same content via the same git servers.

Cluster redundancy

In a true geo-redundant environment, multiple EDA deployments are running in different locations, where one deployment is designated the active, and the other deployment is designated as standby. Both deployments must have the same git servers configured so they have access to the same data.

An operator must define the members of a geo-redundant cluster, whether each member is a standalone EDA deployment configured to be part of a cluster. It takes two members to form a cluster, with manual intervention currently required for switchovers to occur. For details, see Switching the active deployment.

Note: These two concepts are distinct and can be used separately. For example, a single EDA deployment can use multiple git servers so that data is stored redundantly across multiple git servers. You can also deploy two EDA deployments for a redundant cluster with only a single git server (the same one) configured for each deployment. If multiple deployments for a redundant cluster are used, the same git servers must be configured on both deployments.

14.4.1.1 Adding remotes

An operator can enable remote redundancy during initial installation or after installation. All cluster members must be running the same software version.

Example: Initial standalone configuration

The following example shows the initial `EngineConfig` CR fields for the standalone member, `us-west-1`. This resource defines a single member cluster with two git servers, exposed via a load balancer or directly via the address `10.0.0.1` for IPv4, or `2000::101` for IPv6, or is reachable via the domain name `cluster.eda.nokia.com` (which maps to the two IP addresses).

```
apiVersion: core.eda.nokia.com/v1
kind: EngineConfig
metadata:
  name: us-west-1
spec:
  git:
    servers:
      - name: git1
        url: https://git1.eda.nokia.com
        credential: git1-token
      - name: git2
        url: https://git2.eda.nokia.com
        credential: git2
    backup:
      repo: sr/eda/backup
    userStorage:
      repo: sr/eda/user-storage
    apps:
      repo: sr/eda/apps
  cluster:
    external:
      ipv4Address: 10.0.0.1
      ipv6Address: 2000::101
      domainName: cluster.eda.nokia.com
      port: 51101
```

Example: Adding another EDA instance

To grow this cluster, first, install another EDA instance into another Kubernetes cluster. The following sample `EngineConfig` CR is for the new EDA instance, `us-east-2`:

```
apiVersion: core.eda.nokia.com/v1
kind: EngineConfig
metadata:
  name: us-east-2
```

```
spec:
  git:
    servers:
      - name: git1
        url: https://git1.eda.nokia.com
        credential: git1-token
      - name: git2
        url: https://git2.eda.nokia.com
        credential: git2
    backup:
      repo: sr/eda/backup
    userStorage:
      repo: sr/eda/user-storage
    apps:
      repo: sr/eda/apps
  cluster:
    external:
      ipv4Address: 10.0.0.1
      ipv6Address: 2000::101
      domainName: cluster.eda.nokia.com
      port: 51101
    redundancy:
      credential: cluster-cred
      active: us-west-1
      members:
        - name: us-west-1
          address: 10.0.0.2
          port: 55000
        - name: us-east-2
          address: 20.0.0.1
          port: 55001
```

Example

Upon starting the `us-east-2` cluster, it attempts to connect to `us-west-1`, which is not yet currently configured as a cluster member. The attempt to join should fail, with `us-east-2` attempting to form a cluster at a back-off interval. The active cluster is then updated to:

```
apiVersion: core.eda.nokia.com/v1
kind: EngineConfig
metadata:
  name: us-west-1
spec:
  git:
    servers:
      - name: git1
        url: https://git1.eda.nokia.com
        credential: git1-token
      - name: git2
        url: https://git2.eda.nokia.com
        credential: git2
    backup:
      repo: sr/eda/backup
    userStorage:
      repo: sr/eda/user-storage
    apps:
      repo: sr/eda/apps
  cluster:
    external:
      ipv4Address: 10.0.0.1
      ipv6Address: 2000::101
      domainName: cluster.eda.nokia.com
      port: 51101
    redundancy:
      credential: cluster-cred
      active: us-west-1
      active: us-west-1
      members:
        - name: us-west-1
          address: 10.0.0.2
          port: 55000
        - name: us-east-2
          address: 20.0.0.1
          port: 55001
```

This resource describes a two-member cluster, where each member is aware of how to reach each other using the credential, address, and port provided. The address and port values can be a DNS name or IPv4/IPv6 address, and is mapped directly to the `ConfigEngine` resource in each cluster.

The `name` field in the `EngineConfig` resource differs per cluster, and should map to one of the `members` listed.

In this example, the cluster grows from 0 members to 2. Both members must specify the same member as active. In this sample configuration, the previously standalone member remains active.

14.4.1.2 Removing remotes

After installation, you can decommission a remote and reinstall it or remove it entirely. You can remove a remote member even if it is unreachable. You can only remove a member that is a standby, so if you want to remove an active cluster, you should first switchover to a member that is not being removed.

Example

The initial configuration below is for a cluster with three members.

```
apiVersion: core.eda.nokia.com/v1
kind: EngineConfig
metadata:
  name: us-west-1
spec:
  git:
    servers:
      - name: git1
        url: https://git1.eda.nokia.com
        credential: git1-token
      - name: git2
        url: https://git2.eda.nokia.com
        credential: git2
    backup:
      repo: sr/eda/backup
    userStorage:
      repo: sr/eda/user-storage
    apps:
      repo: sr/eda/apps
  cluster:
    external:
      ipv4Address: 10.0.0.1
      ipv6Address: 2000::101
      domainName: cluster.eda.nokia.com
      port: 51101
    redundancy:
      credential: cluster-cred
      members:
        - name: us-west-1
          address: 10.0.0.2
          port: 55000
        - name: us-east-2
          address: 20.0.0.1
          port: 55001
        - name: us-east-3
          address: 30.0.0.1
          port: 55001
```

To update the configuration so there is only a standalone member, `us-west-1`, the following would need to occur:

1. Make `us-west-1` the active member.
2. Remove the `us-east-3` member from `us-west-1` and `us-east-2`.
3. Uninstall `us-east-3`.
4. Remove `us-east-2` from `us-west-1`.
5. Uninstall `us-east-2`.

14.4.1.3 Cluster members

The following fields in the in `EngineConfig` CR define the members of a cluster:

- In the `.spec.cluster.redundancy.members` context:
  - `name` - a user friendly name for the member. This setting is validated against the name of the local `EngineConfig` resource to determine which cluster member the local ConfigEngine is. This requires changes to the current `EngineConfig` name. If no members are provided, the cluster is assumed to be a single member cluster, and the name check does not occur.
  - `address` - either an IPv4 or IPv6 address, or domain name that can be resolved.
- In the `.spec.cluster.redundancy.credential` context: `credential`. This value is used for authentication between members. The value must be the same for all members.
- `port`: the port on which a peer ConfigEngine (proxied through APIServer) is exposed. Both the address and port are external addresses/ports may live on a load balancer.

For a geo-redundant deployment, the following settings apply to members of a cluster:

- The set of git servers provided at `.spec.git.servers` context must be identical.
- The number of replicas for the API server (`.spec.api.replicas`) and State Aggregator (`.spec.stateAggregator.replicas`) must be consistent between the clusters. This ensures that standby clusters can take the load of the active cluster. This check occurs only initially syncing a remote, as the values can change post run-time.
- The content of `.spec.cluster` context must match. This includes members in `.spec.cluster.redundancy.members`, and information around external reachability of the cluster in `.spec.cluster.external` context
- The content of `.spec.playground` and `.spec.simulate` must match.

14.4.1.4 Verifying the geo-redundancy state

Procedure

To verify the state of the geo-redundant members of a cluster, use the EDA toolbox deployed in the EDA Kubernetes cluster to execute the following command:

```
$ edactl cluster
Name       Address     ActivityState  AveLatency(ms)  Reachable  InSyncWithActive
us-east-2  192.0.2.11  Standby                        true       true
us-west-1  self        Active                         true       true
```

14.4.1.5 Switching the active deployment

Prerequisites

Before switching the active deployment, verify that the connectivity between the deployments is as expected. If both deployments are up and running, but there is no connectivity between them, a switchover can cause both deployments to think they are active. This can cause issues.

Procedure

To switch which EDA deployment is active, open the EDA toolbox on the EDA deployment that needs to be made active and execute the following command:

```
edactl cluster take-activity <name of member to make active>
```

Expected outcome

If the other deployment is still active and can be reached, the local deployment instructs it to go into standby mode, and make itself active.

If the other deployment is no longer available (or reachable), the local deployment assumes it to be lost and makes itself active.

14.5 Node management

This section includes topics such as node RBAC and node discover, including bootstrapping and zero-touch provisioning (ZTP).

14.5.1 Node RBAC

EDA supports the use of node RBAC to secure communications between EDA and nodes. System administrators can configure node security profile, node groups and node users, the use of TACACS.

14.5.1.1 Node groups

The NodeGroup resource defines a group on a node. It includes RBAC settings and the selection of services to which users belonging to the group has access, and TACACS configuration. A node group has the following attributes:

- an optional name override in groupName, allowing the resource name and local group name on the target to be different
- the set of enabled services
- an indicator if the group provides superuser permissions
- a set of rules, being target specific RBAC rules
- mapping to a privilege level in the TACACS container

A NodeGroupDeployment resource is used to deploy NodeGroup resources to target toponodes.

Rules

Users of for node groups can define a set of rules that are specific to a specified operating system. The **Rules** section of the NodeGroup resource includes the following parameters that define a rule:

- An action, which can be one of the following:
  - Deny
  - ReadWrite
  - Read
- An operatingSystem - which OS to apply this rule to.
- A match - an OS-specific path, for example interface for SR Linux, or configure port for SR OS.

Rules that match the operating system of the target are deployed to that target.

The default for action is set to ReadWrite, and to simplify the majority of deployments the operatingSystem is set to srl.

Superuser

EDA supports a superuser attribute; if enabled for a node user group, users that belong to the node group can perform all functions on the system, including sudo and root access, if available.

TACACS+

System administrators commonly use TACACS+ to authenticate users, and then use the local device enforce a locally-defined rule set, or role. In EDA, enforcement uses the privilege level in TACACS+. If TACACS+ is used for authentication and if a privilege level is returned, a user is granted the set of permissions from all groups that match that privilege level and lower (following TACACS+ implementation of higher privilege levels inheriting permissions of lower levels).

Note: TACACS+ server configuration is currently done through a Configlet application.

Services

You can select the of services (read: management services like gNMI, NETCONF, CLI) that a group is allowed in the **Services** field. Select one or more of the following services:

- CLI
- FTP
- gNMI
- gNSI
- gRIBI
- Reflection
- JSON-RPC
- NETCONF

Default sudo group

The default `sudo` node group is provided during the bootstrap process or playground deployment. This group enables critical services and provides read/write access to all paths. This `NodeGroup` is referenced by the admin `NodeUser` resource that is provided with playground KPT package.

The following example shows a `sudo NodeGroup` resource:

```
apiVersion: core.eda.nokia.com/v1
kind: NodeGroup
metadata:
  name: sudo
  namespace: eda
spec:
  services:
  - GNMI
  - CLI
  - NETCONF
  superuser: true
```

14.5.1.1.1 Creating node groups

About this task

In the EDA UI, perform this task from the **System Administration** → **Node Management** → **Node Groups** page.

Procedure

Step 1. Click **Create**.

Step 2. Configure metadata for this resource.

Set the following fields:

- **Name**
- **Namespace**
- **Labels**
- **Annotations**

Step 3. Configure specifications for the node group.

- Provide a group name. If the you do not provide name, the system uses the resource name.
- In the **Services** drop-down list, select the services that users who belong to this group can access.
- Enable the **Superuser** field to make members of this node user group superusers.

Step 4. In the **Rules** section, click **Add** to configure rules.

Set the following fields to define the operating system match rule for this group:

- **Action**: select an action from the drop-down list
- **Operating System**: select **srl** for SR Linux or **sros** for SR OS.
- **Match**: a string to match input against; for example, **interface** for SR Linux or **configure port** for SR OS. Rules here should be specified in the target specific format.

Step 5. If TACACS is used for authentication, in the **TACACS** section, select the privilege level.

Step 6. Click **Add To Transaction**.

14.5.1.2 Node users

The `NodeUser` resource defines a node user using the following parameters:

- username and password
- node groups to which the user belongs
- SSH public keys to be deployed for the user

14.5.1.2.1 Creating node users

About this task

In the EDA UI, perform this task from the **System Administration** → **Node Management** → **Node Users** page.

Procedure

Step 1. Click **Create**.

Step 2. Configure metadata for this resource.

Set the following fields:

- **Name**
- **Namespace**
- **Labels**
- **Annotations**

Step 3. Configure the specifications for this node user.

In the **Specification** section, set the following fields:

- **Username**: provide a name for this user. If you do not provide a username, the resource name is used.
- **Password**: provide a password for this user

Step 4. Configure group bindings.

In the **Group Bindings** section, click **Add**.

- Select the toponodes.
  - To use a label selector to select nodes, in the **Node Selector** section, click **Add a Label Selector** .
  - To identify specific nodes, in the **Nodes** section, click **Add item** to select toponodes from the drop-down list.
- In the **Groups** section, click **Add** to specify the node groups to which this user belongs.

Step 5. In the **SSH Public Keys** field, click **Add item** to set the SSH public key to deploy for the user.

Example: `NodeUser` resource

```
---
apiVersion: core.eda.nokia.com/v1
kind: NodeUser
metadata:
  name: node-user
spec:
  username: test
  password: testPassword
  groups:
  - admin
  nodeSelector:
  - eda.nokia.com/role=spine
  - eda.nokia.com/role=leaf
  - eda.nokia.com/role=superspine
  sshPublicKeys:
  - "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQCsYFM9U5hwi+hcZGr5EPjbcneMB
+CRmJ1zYDI5wXq8BgtJdXLCqRnsHtdTpfXn5agwGfkMntsw+/whDjJj3HBX6FxAnB9CO/tHw0AZ7ww
Aagfp5TFkQwGsUVroJlqUfiu1I1yHqNx+etS8DrAAyLtiUMaEvSLztpqjG/4E3TsEvR1pRgt5Ok
EfX7CX8PIuCtKvuFBh7aaU6W8a5kvInQaL0TrxEWHb3cnqwTryRri+ohtHFaSFvpJsTT07in3j2Uw
Pw8ICpi75xd8PMKC8CIqijIACIHMIADK0qUIhB+VEXhFp0RPihXraX8v+l7IFRBHLHqIW8ygJ5PUXQKx6+p
+TRDhCNtuhL7pd+TWFJPqD9bZigIWEYfQE3dQ2ZNabXr+5sOxyeHot1nYUj5TiFLuCtNz36i3TkXNbHfKxu
MymoLEiSOZyD2EkKNlzvxiW4RJl2wZAjjg9pqZILNkbkFVNo0gE3QSkr6fzIRFy27xUBWmG8zi06T4ium
EvmhL05Ri3cPDzWQa4FoI9kzt1iCgCFqhHioK882CjZoWt9vX5+JqqddKXJV7oix5jlTvKtEQBYFSKTra2Mt
+Gwpbn5bG3TtaumtpX4rK9PVPKnfCLccwRnp+mpijxcGA91N7+2Ud9fSPe8JX/jdGfSXAyU1GuCNI/pHjp0ILq
Fy2GwQseGQ== admin"
```

14.5.1.3 Node security profile

The `NodeSecurityProfile` resource provides the parameters that define how to secure communication between EDA and a node. The `NodeSecurityProfile` resource facilitates the configuration, generation, and rotation of TLS certificates, trust bundle management, and secure communication with specified nodes.

Node selection

In `NodeSecurityProfile` resource, you can select nodes using the following methods:

- listing the nodes: in the `nodes` field, list the `TopoNodes` to which the profile applies
- by label: in the `nodeSelector` field, select a label that applies to `TopoNodes` that meet the criteria selected. This field can contain a list of label selectors; a `TopoNode` must contain at least one of the labels to inherit the profile's settings.

A nodeSelector set to an empty string ("") means that the profile applies to all nodes.

The nodes field takes precedence over the nodeSelector setting. If multiple profiles match a node's labels, the profile whose name is first in alphabetic order is applied.

TLS configuration

The tls context indicates whether the connection to the node is secure (with TLS) or insecure (without TLS). The absence of the tls field implies an insecure connection, while its presence signals a secure connection.

EDA-managed certificates

When EDA is responsible for managing node certificates, the tls context must include the following entries:

- issuerRef: a reference to a CertManager Issuer, which is responsible for issuing the certificates.
- csrParams: the Certificate Signing Request (CSR) parameters define the parameters for certificate generation and rotation.
  - csrSuite: the key and digest set to be used for generating the CSR.
  - commonName: the common name (CN) to include in the certificate. This value is auto-generated.
  - country: the legally registered country of the organization.
  - state: the state or province where the organization is located.
  - city: the city in which the organization is based.
  - org: the name of the organization requesting the CSR.
  - orgUnit: the department or division within the organization requesting the certificate.
  - certificateValidity: the duration for which the certificate remains valid post-issuance.
  - SAN (Subject Alternative Names):
    - dns: List of DNS names used to access the node.
    - emails: Email addresses associated with the certificate.
    - ips: IP addresses that the certificate should validate.
    - uris: Specific URIs that the certificate needs to authenticate.

The following is an example of nodeSecurityProfile where EDA manages certificates:

```
apiVersion: v1
items:
  - apiVersion: core.eda.nokia.com/v1
    kind: NodeSecurityProfile
    metadata:
      annotations:
        config.k8s.io/owning-inventory: 8c2644abc6befe73d5ad0cfc386ec155f31bc07d-
1729769165484669347
        kubectl.kubernetes.io/last-applied-configuration: >
          {"apiVersion":"core.eda.nokia.com/v1","kind":"NodeSecurityProfile","metadata":
{"annotations":{"config.k8s.io/owning-inventory":"8c2644abc6befe73d5ad0cfc386ec155f31bc07d-
1729769165484669347"},"name":"insecure","namespace":"default"},"spec":{"nodeSelector":
["eda.nokia.com/security-profile=insecure"]}}
      name: insecure
      namespace: eda
    spec:
      nodeSelector:
        - eda.nokia.com/security-profile=insecure
  - apiVersion: core.eda.nokia.com/v1
    kind: NodeSecurityProfile
    metadata:
      annotations:
        config.k8s.io/owning-inventory: 8c2644abc6befe73d5ad0cfc386ec155f31bc07d-
1729769165484669347
        kubectl.kubernetes.io/last-applied-configuration: >
          {"apiVersion":"core.eda.nokia.com/v1","kind":"NodeSecurityProfile","metadata":
{"annotations":{"config.k8s.io/owning-inventory":"8c2644abc6befe73d5ad0cfc386ec155f31bc07d-
1729769165484669347"},"name":"managed-tls","namespace":"default"},"spec":{"node
Selector":["eda.nokia.com/security-profile=managed"],"tls":{"csrParams":{"certificate
Validity":"2160h","city":"Sunnyvale","country":"US","csrSuite":"CSRSUITE_X509_KEY_TYPE_RSA_
2048_SIGNATURE_ALGORITHM_SHA_2_256","org":"NI","orgUnit":"EDA","state":"California"},"issuer
Ref":"eda-node-issuer"}}}
      name: managed-tls
      namespace: eda
    spec:
      nodeSelector:
        - eda.nokia.com/security-profile=managed
      tls:
        csrParams:
          certificateValidity: 2160h
          city: Sunnyvale
          country: US
          csrSuite: CSRSUITE_X509_KEY_TYPE_RSA_2048_SIGNATURE_ALGORITHM_SHA_2_256
          org: NI
          orgUnit: EDA
          state: California
        issuerRef: eda-node-issuer
  - apiVersion: core.eda.nokia.com/v1
    kind: NodeSecurityProfile
    metadata:
      annotations:
        config.k8s.io/owning-inventory: 8c2644abc6befe73d5ad0cfc386ec155f31bc07d-
1729769165484669347
        kubectl.kubernetes.io/last-applied-configuration: >
          {"apiVersion":"core.eda.nokia.com/v1","kind":"NodeSecurityProfile","metadata":
{"annotations":{"config.k8s.io/owning-inventory":"8c2644abc6befe73d5ad0cfc386ec155f31bc07d-
1729769165484669347"},"name":"unmanaged-tls","namespace":"default"},"spec":{"nodeSelector":
["eda.nokia.com/security-profile=unmanaged"],"tls":{"trustBundle":"eda-node-trust-bundle"}}}
      name: unmanaged-tls
      namespace: eda
    spec:
      nodeSelector:
        - eda.nokia.com/security-profile=unmanaged
      tls:
        trustBundle: eda-node-trust-bundle
kind: List
metadata:
  resourceVersion: ""
```

External certificate management

If certificates are managed outside of EDA, the `tls` section must reference an external trust bundle. The `trustBundle` field in (ConfigMap) contains a reference to a ConfigMap that holds a CA certificate. EDA uses this CA certificate to verify the node’s certificate whenever it establishes a connection. The trust bundle must be provided if node certificate management is performed outside of EDA, allowing the node to validate certificates through an external authority.

```
apiVersion: core.eda.nokia.com/v1
kind: NodeSecurityProfile
metadata:
  name: example-node-security-profile
spec:
  nodeSelector:
    - "eda.nokia.com/role=leaf"
  tls:
    trustBundle: "node-trust-bundle"
```

14.5.2 Node discovery

This section includes topics related to initial node discovery.

14.5.2.1 Bootstrapping

The `Init` application generates an initial configuration file for nodes that require bootstrapping. The input to the `Init` application is an `Init` resource, which specifies which toponodes to select and for which toponodes to create an initial configuration.

The initial configuration file is stored in the artifact server. When toponodes connect to NPP, NPP pushes the initial configuration file to the node.

Additionally, the `Init` application generates the Python provisioning script for SR Linux and bootstrap files needed for SR OS or other operating system. Based on the same selection criteria, a bootstrap file or Python provisioning script is generated for the selected nodes.

The bootstrap file or Python provisioning script ensures that the node boots into the version specified in the toponode. The software and any other artifacts are downloaded to the node during ZTP using HTTPS.

By default, if no toponode selectors are present in the `Init` resource, an initial configuration file is generated for every toponode present in EDA.

Management interface IP address assignment

The `Init` resource allows you to configure the management interface IP assignment method using DHCP or by defining static IP addresses. For details, see Enabling DHCP clients and Setting static management IP addresses.

Saving node configuration on commit

To specify whether the node configuration is saved after each commit or not, in the `Init` CR include the entry `commitSave: true`. The `Init` script must reflect the `commitSave` value in the generated initial configuration.

14.5.2.1.1 Preparing for bootstrapping

Ensure that you meet the following requirements:

- The NodeSecurityProfile resource (for TLS) must be configured to ensure successful onboarding. For details, see Node security profile.
- A node requires the relevant `EDA-CE license` resource to be applied. Without this license, the node may not move to READY state.

  ```
  apiVersion: core.eda.nokia.com/v1
  kind: License
  metadata:
    name: eda-license
    namespace: eda-system
  spec:
    enabled: true
    data: "ACoAgOlJq7AABoAU6V6W6XAERezbcYa+ZRZLg8M5Iyq
  MgAABAATAEVEQS1bQkNdLTAuMC4qAAACABIATm9raWEuY29tLOVEQQAAAMAAMQCjorJ
  +SPKP3if9pcD3OhqlyaWK1VE89JWreOWkyOJcbIWO602C+iwp
  +FFp8AwAAAADAB4ARWRhIGxpY2Vuc2UgSW50ZXJuYWwgVGVzdAAAAAUAHADl0z
  NnAAAAAABgKWcAAAAAADohaAAAAADAACQAoKr6XCCQCZj1rWFYik1dGbiqG7TWRK2orh
  +0sjUKXNYBACkAMDAwMDAwMDAtMDAwMC0wMDAwLTAwMDAtMDAwMDAwMDAwMDAwAAAAAAYADAABAAQAAAAAAMAAMAC/
  KQqX7Di/m1d0zYz9quIyghaHatF0yDvDgK/fFr011Wa/7FN3LO/OoD3a
  Hg8AXQFkOEh6ejQrTlFyNmVJNTNsVW9SMi9JV2xXd1NqMUF3QVh0eEd6LzhGdlp0WXph
  TkdOQ1RWRnNCQ3wwZ0p2b21pSDNiZHFTSFBYQ2R6d0xxVlNhM3FZZUZuL1BGMnho
  SjN6OS8yS3RlVGpmUngreWFNS1NwZ0p5OE12YlBVbmw2TUFpNHRXR1g4U3R0WXFBN21uVUNhVHp5e
  XpLOWtXcWgwZVZtR1oyV09RTURML0thaWY1RGMva21tc0NVY042RUdNZUNi
  TmdvV2RKUFlXZ1o4c2hlaG03b2tsZHdsSDBxMXZWdjhHMjZ4OVUxbTd2ellBN3BDNkFXODJyZ3FsaExWTUJx
  Ym11VDdKSzdPWWhzYVp4Q3h4a2lIbWZ5KytNY3FLVHFBUk1McWhYRzRIb290ME0xK1RaRVZTdUJKNFl5a3pke
  HdVV3pGZGRZdjg5Ym5uUHBsdXc9PQAAAAA="
  ```

- If the deployment uses EDA DHCP for bootstrapping, `GlobalConfig` and `UdpProxy` CR may be needed.

  Following is an example of `GlobalConfig` resource

  ```
  apiVersion: v1
  items:
    - apiVersion: core.eda.nokia.com/v1
      kind: GlobalConfig
      metadata:
        name: global
        namespace: eda-system
      spec:
        dhcp:
          domainName: mv1-3.dclab.nuq.ion.nokia.net
          httpPort: 9200
          httpsPort: 9443
          ipv4Address: 10.11.12.13
          ipv6Address: 3001:cafe:11::2
  kind: List
  metadata:
    resourceVersion: ""
  ```

  Following is an example of a `UdpProxy` resource:

  ```
  apiVersion: v1
  items:
    - apiVersion: core.eda.nokia.com/v1
      kind: UdpProxy
      metadata:
        annotations:
          config.k8s.io/owning-inventory: aeb8a5709fd9a90c89d3d3dcc1d9c3817f2618ae-
  1732279916926223978
          kubectl.kubernetes.io/last-applied-configuration: >
  ```

```
          {"apiVersion":"core.eda.nokia.com/v1","kind":"UdpProxy","metadata":
{"annotations":{"config.k8s.io/owning-inventory":"aeb8a5709fd9a90c89d3d3dcc1d9c3817f2618ae-
1732279916926223978"},"name":"eda-dhcp","namespace":"eda-system"},"spec":{"buffer
Size":65535,"destHost":"eda-dhcp","destPort":67,"idleTimeout":60,"proxyPort":67}}
      name: eda-dhcp
      namespace: eda-system
    spec:
      bufferSize: 65535
      destHost: eda-dhcp
      destPort: 67
      idleTimeout: 60
      proxyPort: 67
  - apiVersion: core.eda.nokia.com/v1
    kind: UdpProxy
    metadata:
      annotations:
        config.k8s.io/owning-inventory: aeb8a5709fd9a90c89d3d3dcc1d9c3817f2618ae-
1732279916926223978
        kubectl.kubernetes.io/last-applied-configuration: >
          {"apiVersion":"core.eda.nokia.com/v1","kind":"UdpProxy","metadata":
{"annotations":{"config.k8s.io/owning-inventory":"aeb8a5709fd9a90c89d3d3dcc1d9c3817f2618ae-
1732279916926223978"},"name":"eda-dhcp6","namespace":"eda-system"},"spec":{"buffer
Size":65535,"destHost":"eda-dhcp6","destPort":547,"idleTimeout":60,"proxyPort":547}}
      name: eda-dhcp6
      namespace: eda-system
    spec:
      bufferSize: 65535
      destHost: eda-dhcp6
      destPort: 547
      idleTimeout: 60
      proxyPort: 547
kind: List
metadata:
  resourceVersion: ""
```

- The init and relevant images must be downloaded to the artifacts server.
  The following resource must be present:

```
apiVersion: v1
kind: Secret
metadata:
  name: srl-node-cred
  namespace: eda
type: Opaque
data:
  username: YWRtaW4=
  password: Tm9raWFTcmwxIQ==
---
apiVersion: v1
kind: Secret
metadata:
  name: srl-ftp-cred
  namespace: eda
type: Opaque
data:
  username: ZnRwdXNlcg==
  password: U2ghbmluZyR0YXIxIQ==
---
apiVersion: artifacts.eda.nokia.com/v1
kind: Artifact
metadata:
  name: srlinux-24.10.1-492
  namespace: eda
```

```
spec:
  repo: images
  filePath: srl.bin
  remoteFileUrl:
    fileUrl: ftp://10.10.10.10/eda/srl_images/srlinux-24.10.1-492.bin
  secret: srl-ftp-cred
---
apiVersion: artifacts.eda.nokia.com/v1
kind: Artifact
metadata:
  name: srlinux-24.10.1-492-md5
  namespace: eda
spec:
  repo: images
  filePath: srl.bin.md5
  remoteFileUrl:
    fileUrl: ftp://10.10.10.10/eda/srl_images/srlinux-24.10.1-492.bin.md5
  secret: srl-ftp-cred
```

14.5.2.1.2 Enabling DHCP clients

To enable the IPv4 and IPv6 DHCP clients on the management interface, in the `Init` resource, include the following entries in the `mgmt` context:

`ipv4DHCP: true`

`ipv6DHCP: true`

In the `mgmt` section, by default, both `ipv4DHCP` and `ipv6DHCP` are set to `true`. Optionally, you can also set the IP MTU, as shown in the following example:

```
apiVersion: bootstrap.eda.nokia.com/v1alpha1
kind: Init
metadata:
  name: init-config
spec:
  nodeSelector:
  - 'eda.nokia.com/role=leaf'
  - 'eda.nokia.com/role=spine'
  - 'eda.nokia.com/role=borderleaf'
  - 'eda.nokia.com/role=superspine'
  - 'eda.nokia.com/role=backbone'
  mgmt:
    ipv4DHCP: true
    ipv6DHCP: true
    ipMTU: 9000
```

14.5.2.1.3 Setting static management IP addresses

To set the management IP address statically, the `init` script must use the `productionAddress` setting from the `Toponode` resource as the IPv4 or IPv6 address in the generated configuration.

The `init` script sets the address as either IPv4 or IPv6 and sets the prefix length.

The table below displays the different combinations of `ipv4DHCP`, `ipv6DHCP` and `productionAddress` settings and the corresponding resulting initial configuration.

Table 26: Combinations for `ipv4DHCP`, `ipv6DHCP` and `productionAddress` settings

Init resource	TopoNode setting	Result
`ipv4DHCP: true`	*	The management interface IPv4 client is enabled in the initial configuration.
`ipv6DHCP: true`	*	The management interface IPv6 client is enabled in the initial configuration.
`ipv4DHCP: false`	IPv4 productionAddress is set	The production address is set as the IPv4 address of the management interface in the initial configuration.
`ipv4DHCP: false`	IPv6 productionAddress is set	The IPv4 address is left unset in the initial configuration and the IPv4 DHCP client is not enabled.
`ipv6DHCP: false`	IPv4 productionAddress is set	The IPv6 address is left unset in the initial configuration and the IPv6 DHCP client is not enabled.
`ipv6DHCP: false`	IPv6 productionAddress is set	The production address is set as the IPv6 address of the management interface in the initial configuration.
`ipv4DHCP: false` `ipv6DHCP: false`	productionAddress is not set	Results in than error: add productionAddress to TopoNode or enable a DHCP client.

Static Routes

To define the static routes in the `Init` CR, specify an IP prefix and a next hop. The `Init` script adds the static routes to the management network instance. For example:

```
apiVersion: bootstrap.eda.nokia.com/v1alpha1
kind: Init
metadata:
  name: init-config
spec:
  nodeSelector:
  - 'eda.nokia.com/role=leaf'
  - 'eda.nokia.com/role=spine'
  - 'eda.nokia.com/role=borderleaf'
  - 'eda.nokia.com/role=superspine'
  - 'eda.nokia.com/role=backbone'
  mgmt:
    ipv4DHCP: true
    ipv6DHCP: true
    ipMTU: 9000
    staticRoutes:
      - prefix: 10.10.0.0/16
        nextHop: 172.16.255.29
      - prefix: 2001:10:10::/64
        nextHop: "200::"
```

14.5.2.2 Zero-touch provisioning

Zero Touch Provisioning (ZTP) allows for a device to be installed in a rack, powered on, and without any additional input from an operator, boot up pull down the software version of its operating system, an initial configuration and any other boot artifacts required for it to be managed.

Most ZTP implementations rely on DHCP to provide an IP address to the DUT and use DHCP options to inform the DUT of the location of any boot artifacts it requires to complete its ZTP process. In SR Linux, the DHCP server provides the URL of a Python provisioning script which is then used by the DUT to perform actions such as software upgrade and applying an initial configuration. In SR OS, the DHCP server provides a URL to a provisioning file which is a text file containing URLs to software images and configuration files.

For devices running SR OS and SR Linux, the devices send a DHCP Discover message with option 61 (client-id) set to the chassis serial number. This setting is used on the DHCP server to associate a DHCP discover message with a specific DUT and allows for the DHCP server to allocate static DHCP leases (IPs) and potentially device-specific boot artifacts (Python script or boot file).

EDA supports the following modes of operation for DHCP aspect of ZTP:

- Use of an internal DHCP server (hosted and managed by EDA)
- Use of an external DHCP server (hosted and managed outside of EDA)

To serve the boot artifacts (Python script, boot file, software, or any other files needed during the bootstrapping process), an artifact server must be present in EDA. An intent is used to allow for artifacts to be added to the server, which is then be retrieved by the devices during boot.

DHCP server

In deployments that use EDA to handle ZTP in its entirety, a DHCP server is required to provide IP addresses to devices.

When a device issues a DHCP discovery message, the client-id option (61) attribute includes their chassis serial number. This serial number is used to associate real devices with node objects in EDA. Additionally, an IP address is assigned to device via a `Target` object. The DHCP server must support the following capabilities:

- Static lease assignment using the client-id (option 61) as the binding between an IP address and a device
- Ability to receive DHCP packets from a DHCP relay (the DHCP relay between the devices and the DHCP server)
- When providing an IP address to the device, the DHCP server must be able to populate option 66 or 67 in the DHCP offer. This option provides HTTPs. The URL points to the ZTP provisioning script or boot file hosted on the artifact server.
- Ability to populate other options as required by the operator, for example:
  - Router option 3
  - Time Server option 4
  - Name Server option 5
  - Domain Server option 6

  - Log server option 7
- Support both IPv4 and IPv6 IP addressing

14.6 Draining traffic

About this task

Draining is the concept of gracefully reducing traffic on a device to reduce the risk and reduce the impact of some planned or unplanned activity. Common examples include:

- Removing a device from service to perform a maintenance activity, such as an upgrade.
- Mitigating traffic loss during a brown out or similar event, that is, a fabric module failure in a system that has no fabric redundancy.

In EDA, draining uses a `Drain` resource to select the default routers to drain traffic from default routers. The `Drain` resource:

- modifies routing policies to ensure that the selected default router is used only for terminating traffic, that is, all traffic that has another route would use those other routes (unless they were also being drained)
- generates an alarm that a drain is present on the default router
- generate a `DrainState` resource to update the set of that nodes where the `Drain` resource is present.

Procedure

You can set the following attributes in the `Drain` resource to select the target default routers using label selectors or by listing the target routers:

- `defaultRouterSelector`: specify a label
- `defaultRouters`: list the `DefaultRouters` resources

Example

```
apiVersion: routing.eda.nokia.com/v1alpha1
kind: Drain
metadata:
    name: drain-redundancy-group-a
    namespace: eda
spec:
   defaultRouterSelector:
   - 'eda.nokia.com/redundancy-group=a'
```

Note: After a `Drain` resource is applied, the system raises the relevant Drain Active alarms, which are cleared upon deleting the drain.

Customer document and product support

Customer documentation

Customer documentation welcome page

Technical support

Product support portal

Documentation feedback

Customer documentation feedback