# Event Driven Automation

Release 25.12

## User Guide

3HE 22348 AAAA TQZZA
Edition: 1
December 2025

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.

---

# 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 to 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.

## 1.2 What's new

This section lists the changes that were made in this release.

*Table 1: What's new in 25.12.1*

| Feature | Location |
|---|---|
| **GUI basics** | |
| You can now view resource related targets. | Related targets |
| You can now pause UI streaming using the Live/pause selector. | Home page |
| Multi-row actions are now supported for deleted resources. | Multi-row actions |
| The Space key is now available for row selection. | Table 7: Data grid controls |
| You can now use the split view panel to switch between YAML and JSON views. | Split view panel |
| Alarm counters are now displayed in resource data grids. | Counters |
| Deviation counters are now displayed in resource data grids. | Counters |
| **Dashboard Designer** | |
| User-created dashboards can now be shared. | • Dashboard designer<br>• Dashboards page |
| **Namespaces** | |
| Enhancements were made for the consistent handling of base namespaces. | Namespaces |
| **Resources** | |
| The UI now displays resource topologies. | Resource topologies |
| **Workflows** | |
| You can now stream logs from workflow executions using the **edactl** command. | Managing workflows with edactl |
| Workflows now support bulk selection of resources. | Triggering a workflow from the resource action menu |
| You can now download workflow artifacts for some workflows. | Workflow Summary page |
| The system now supports duplicating a workflow action. | Running a workflow from the Workflow Definition List page |
| **Alarms** | |
| Alarms lists are now available for Last Acknowledged By, Last Suppressed By, and Targets Affected. | The Alarms list |
| Alarm quick filters are now available. | The Alarms list |
| You can now navigate from the Information panel JS Paths to Query Builder. | The Alarms Information panel |

| Feature | Location |
|---|---|
| **EDA Query Language** | |
| The regex comparison operator (~) is now supported with the Where clause. | Elements of a query |
| Support is now available for "is set" and "is not set" with the Where clause. | Elements of a query |
| The concat() function is now supported. | Elements of a query |
| **Transactions** | |
| • Transaction history and logs have been simplified.<br>• The **Transaction Topology** page displays all of the input and output resources that are involved in a transaction.<br>• You can now re-add transaction input CRs to the basket.<br>• Input resource only transaction results can now be displayed in the UI.<br>• Basic and advanced result filtering is now available in the UI.<br>• Dry-run revert and restore is now supported from the UI.<br>• Diffs grouping is now supported.<br>• Transactions result types have been simplified.<br>• A revert fails when a resource has more recent changes<br>• Transactions now automatically revert on failure.<br>• Transaction state status bar now displays in the UI. | • Transactions<br>• Transactions page |
| **Topology** | |
| You can now toggle between the horizontal and vertical layout. | The Topologies page |
| **EDA applications** | |
| In the UI, the EDA Store page now includes sorting capabilities. | EDA Store |
| The app page in the UI now includes:<br>• workflow and resource definitions that come with the app<br>• supported operating systems, the versions supported, and if support is alpha or beta<br>• tabs that provide markdown information for apps that support them | Apps |
| The system now supports dry-runs for app installation, uninstallation, and updates. | • Installing an application<br>• Uninstalling an app<br>• Upgrading apps |

| Feature | Location |
| --- | --- |
| You can now update settings for installed apps that have them. | Editing app settings |
| App settings are now shown in the **Summary** page during installation. | Installing an application |
| **Security** | |
| The **User** page now displays the federation provider for federated users. | • Remote directories<br>• Users |
| The **User** page now displays user log in information, including last successful login and number of failed logins since the last successful login. | Users |
| Removed a note indicating that imported federated users must be deleted explicitly when a federation is deleted | Deleting a federation |
| Enhancements have been made for inferring table rules from resource rules. | Rules |
| **Administration** | |
| You can now reset a user's storage. | Restoring a user's default persistent settings |

# 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.

### EDA store

EDA uses a variety of supporting applications to manage its wide range of capabilities. EDA uses an on-product EDA Store to display the set of available EDA applications, and assist in their installation, update, and removal. The EDA 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 Twin

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 Twin, a containerized virtual infrastructure that emulates the production network by creating its digital twin. EDA uses the Digital Twin to validate intent across the life cycle of the fabric. The Digital Twin is also an essential part of the network validation phase of the CI/CD pipeline process. The EDA framework integrates the Digital Twin 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 Twin 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 Twin 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 Twin 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 Twin 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 Twin 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.

#### Fluent Bit and Fluent Operator

EDA uses Fluent Bit to assist with the processing of logging data. EDA uses a Fluent Bit pod on each Kubernetes worker node to collect logs from the EDA microservices to collect logs from the EDA microservices and one additional Fluent Bit pod as a log aggregator. By default, the Fluent Bit aggregator writes API access logs in their own file, logs each microservice data in their own directories, and maintains an aggregate errors log.

Fluent Operator manages the Fluent Bit configuration and simplifies adding output destinations such as a syslog server.

### 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 intended 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 intended 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:

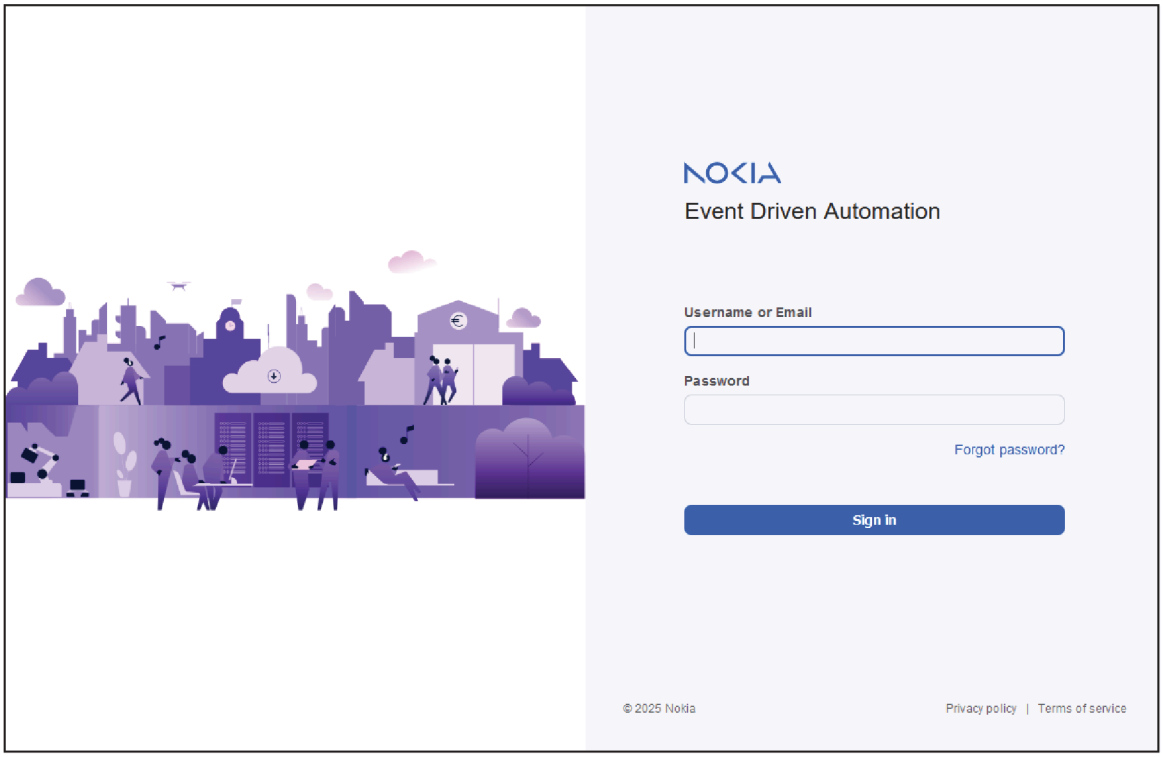
- the URL to the EDA server; for example, https://<server URL>:9443/
- a valid user account and password for EDA

### Procedure

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

**Note:** The default port for EDA is 9443, but consult your local administrator for your organization's URL for EDA.

*Figure 1: The EDA login page*




*sc0203*

**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

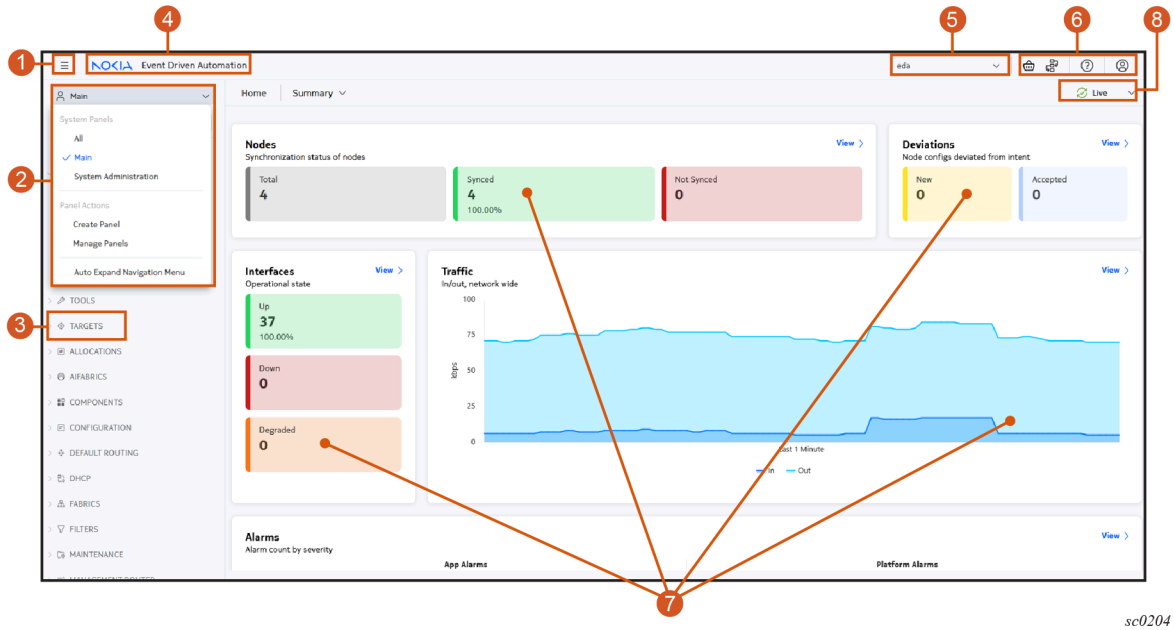
The EDA **Home** page is the first landing page for the EDA GUI. While some elements of the **Home** page are unique, it also includes a set of standard controls that are available from all pages within the EDA GUI.

The EDA **Home** page includes a drop-down that allows you to choose between two views:

- the **Summary** page (default), that displays status summaries for nodes, interfaces, and traffic as well as active alarms
- the **Platform Status** page, that displays summary information for the status of key EDA component

## The Summary page

*Figure 2: The EDA Summary page*




sc0204

*Table 2: 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 controls | Use these tools to:<br>• Change between:<br>– the **Main** EDA menu, providing access to network management functions<br>– the **System Administration** menu, providing access to functions that manage EDA itself<br>– **All**, which displays all menu selections for the Main and System Administration menus<br>• Enable **Auto Expand** for the navigation menu, which causes a collapsed menu to expand horizontally when you hover over any menu icon<br>• Search the full menu items for items that match specific text<br>• Expand or collapse all menu categories<br>**Note:** Because EDA allows you to create your own custom navigation panels, additional panels may be available for selection here. These could be panels you have created |

| # | Name | Function |
|---|---|---|
| | | yourself, or created by others and then published for sharing with other users.<br>. |
| 3 | Menu category | Click to expand or collapse a category to reveal or hide the individual links that are grouped within it. |
| 4 | Event Driven Automation Home button | Click here on any page of the EDA GUI to navigate to, or re-load, the **EDA Home** page. |
| 5 | Namespace selector | Use this drop-down selector to choose a working namespace: either all namespaces, or one specific namespace. This selection affects the namespace from which to display data, and either create or manage resources.<br>  **Note:** The namespaces listed in the selector are limited to those namespaces that you have permission to access. |
| 6 | Common buttons | • The **Transaction** basket: indicates the number of pending resource changes for the current user. Click to open the **Transactions** form.<br>• The **Workflows** button opens the **Workflow Executions** form, which displays recent workflows and their status.<br>• The **Help** button: click to open a menu to access API documentation, hotkey configuration, and Release information.<br>• The **User settings** button: click to open the **User Settings** menu. |
| 7 | Dashlets | Each dashlet displays important information about the status of the EDA application and the network it is managing.<br>Clicking the View link in any dashlet opens the EDA GUI page specific to that dashlet's information.<br>  **Note:** All dashlets are "live"; they continuously update to show the latest data. |
| 8 | Live/Pause selector | Use this drop-down selector to start or pause streaming updates on all resource pages, including the following:<br>• datagrids<br>• dashboards<br>• schema forms<br>• topologies<br>Clicking **Pause** pauses the stream of data which can be helpful for pages with high rates of change.<br>Clicking **Live** resumes the streaming of data. |

| # | Name | Function |
|---|---|---|
| | | **Note:** While paused, changing the filter on an alarm or resource data grid fetches the latest data.<br><br>**Note:** Navigating to a new page resumes streaming. |

The following default dashlets display on the **Summary** page:

- **Nodes**: displays the synchronization state of the nodes known to EDA (total nodes: synced nodes and unsynced nodes).
  Clicking the **View** link from this dashlet takes you to the **Nodes** list.
- **Deviations**: displays the number of nodes that are configured in a way that differs from the last intent known to EDA. Separate counts are displays for those deviations that have been accepted (incorporated into the stored intent) and those that have been detected but have not been accepted.
  Clicking the **View** link from this dashlet takes you to the **Deviations** list.
- **Interfaces**: displays the operational state of the interfaces known to EDA (Up interfaces, Down interfaces, Degraded interfaces).
  Clicking the **View** link from this dashlet takes you to the **Interfaces** list.
- **Traffic**: displays total inbound and outbound traffic for the network as a whole.
  Clicking the **View** link from this dashlet takes you to the EDA Query Builder, displaying the sum of in and out traffic rates for all nodes: `v.namespace.node.srl.interface.traffic-rate fields [sum(in-bps), sum(out-bps)]`.
- **Alarms**: displays the number of current app alarms and platform alarms, and their percentage distribution by alarm type.
  Clicking the **View** link from this dashlet takes you to the **Alarms Summary**.

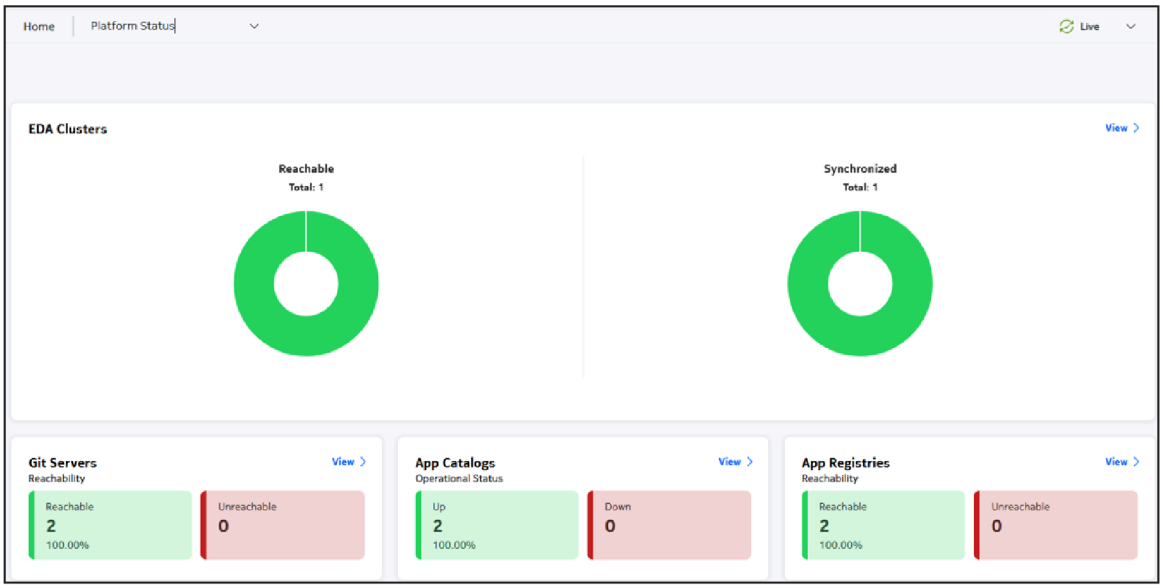
**Related topics**

*Transactions*
*User settings*
*Information panel*

## 3.4 The Platform Status page

The second view available from the **Home** page of the EDA GUI is the **Platform Status** page.

*Figure 3: Platform status*




*sc0205*

Clicking the **View** link from any dashlet opens the **Alarms List**.

*Table 3: Elements of the Platform Status page*

| Dashlet | Description |
|---|---|
| EDA clusters | When configured for Geographic Redundancy, EDA maintains separate instances on independent clusters so that the backup cluster can take over if the primary fails. The primary and backup EDA clusters regularly synchronize so that the latest data is still being used after a switchover to the backup cluster.<br>This dashlet indicates whether clusters are reachable, and whether the reachable clusters are correctly synchronized. |
| Git servers | Shows the reachability status of EDA's Git servers. These servers are used for persistent storage of resources, installed apps, and user settings. |
| App catalogs | An app catalog is a structured Git repository that contains all information about an app, including where to find the app image containers. |
| App registries | An app registry is an OCI-compliant container registry, and contains the actual app image containers. |

## 3.5 GUI elements

The following elements are available at the upper right or right edge of every page within the EDA GUI. This section also contains recurring elements that appear on multiple pages within the EDA GUI.

### 3.5.1 Transactions

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 4: 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*

### 3.5.2 Workflows

A workflow is a series of steps required to perform a task or process, such as performing a ping or performing a route lookup.

The **Workflows** button allows you to interact with workflows from anywhere in the EDA UI.

*Figure 5: Workflows button*

When you click the **Workflows** button, the 10 most recent workflows that you executed are displayed, along with any related notifications, such as if a workflow has completed or is waiting for input. From here, you can:

- click one of the workflow executions to take you to its **Summary** page.
- click the **List** button to take you to the **Workflow Executions** page.

**Related topics**

*Workflows*

*Workflow Executions page*

*Workflow Summary page*

### 3.5.3 User settings

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

*Figure 6: User settings button*

- **User information**: displays the following user and login information:
  - name of the currently logged-in user
  - role of the user
  - date and time of the last successful login
- **Appearance Theme**: click to select a display theme from among the following:
  - Follow system theme: ignore the EDA theme settings, and instead adopt a light or dark theme based on your system settings.
  - 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
- **High Contrast Charts**: controls the color selection for chart segments; enabling high-contrast charts makes EDA charts easier to read for colorblind users.
- **Change Password**: click to change the password for the current user. You must re-authenticate before you can complete the password change.
- **Sign Out**: click to sign out of the EDA application.

### 3.5.4 Help

Click to access the following available information:

- API Documentation: opens the **API Documentation** page.

  **Note:** From the **API Documentation** page, you can download the API documentation as a zip file.

- Hotkeys: opens a form displaying the keyboard hotkeys available for EDA actions.

  **Note:** You can also see the list of hotkeys by pressing "Shift-?" on your keyboard.

- Release Information: opens a form displaying the EDA release number and version.

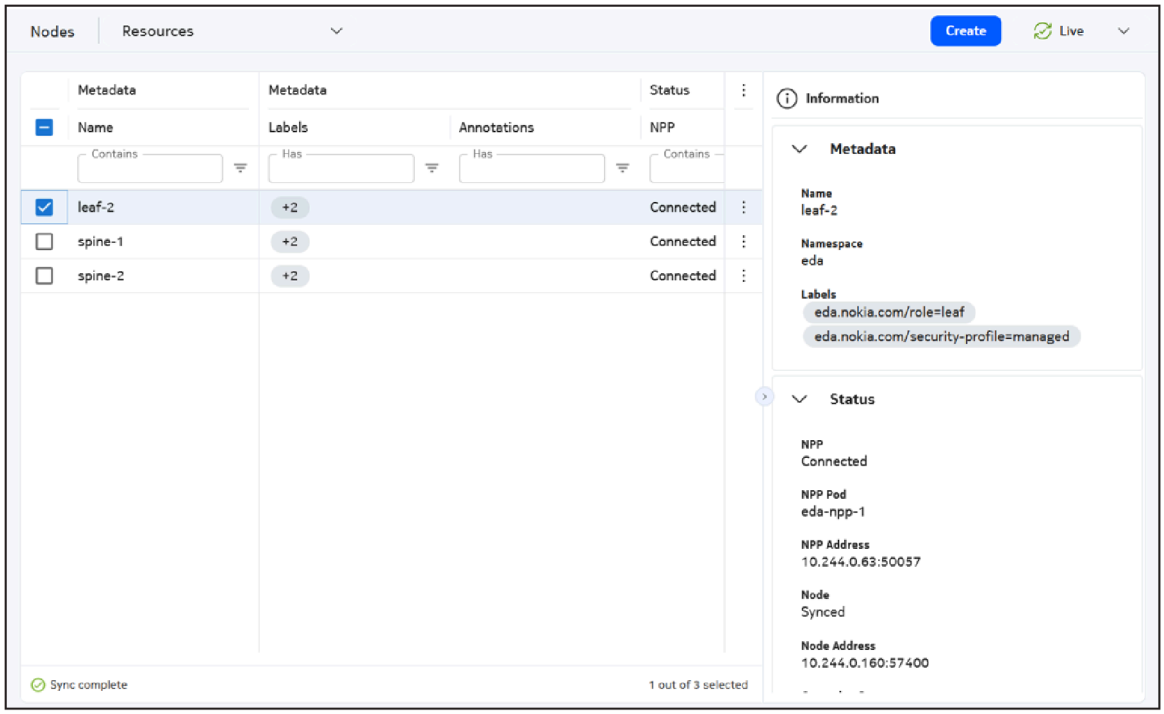
### 3.5.5 Information panel

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.

To enhance readability, fields that would contain no information are excluded from the information panel.

*Figure 7: An example of an expanded information panel, showing the collapse control*



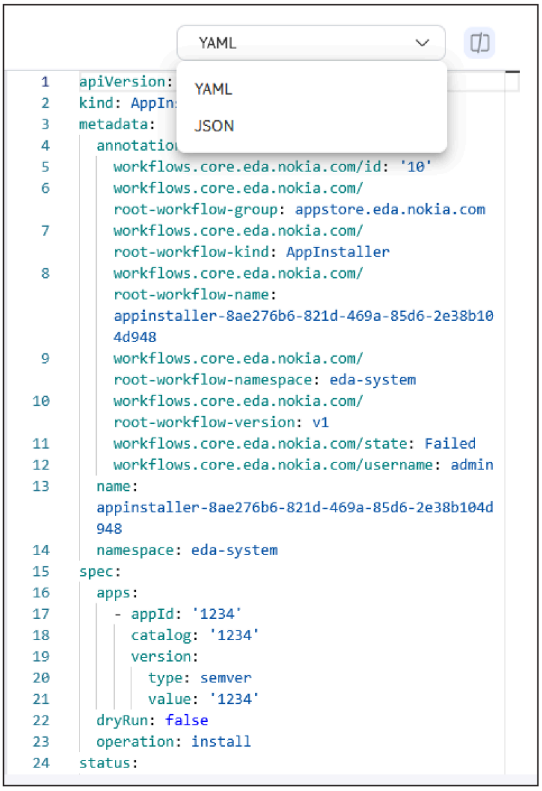

*sc0206*

### 3.5.6 Split view panel

When viewing the details of an element in the EDA GUI, such as a node or a workflow, the **Details** page opens and displays outline, status, and configuration information for the element. The **Split view** panel displays configuration information that can be viewed in either YAML or JSON format.

You can switch between YAML and JSON format by clicking the drop-down menu and selecting the format.

*Figure 8: Split view panel*




sc0304

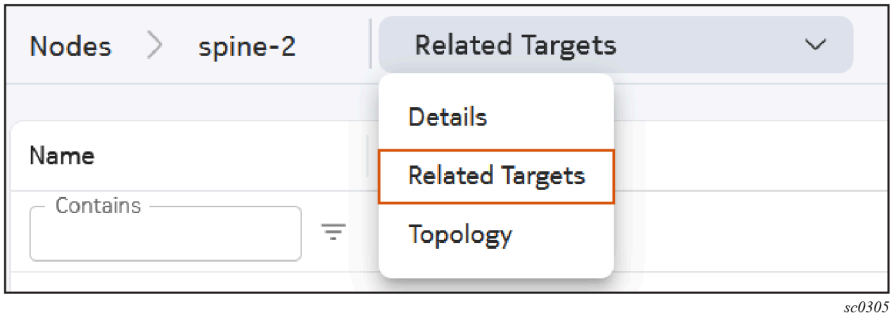
## 3.5.7 Related targets

When viewing a resource in the EDA UI, the **Related targets** view allows you to view a list of targets where the selected resource's intent generated a part of the target's configuration.

**Note:** The **Related targets** view does not account for read relationships between resources. For example, BGP group resources do not have any related targets because the BGP configuration of the target is generated by a different resource intent which reads the BGP group resource as an input.

You can switch to the **Related targets** view by selecting **Related targets** from the drop-down list at the upper right of the page.

*Figure 9: Related targets view*




## 3.6 Key bindings

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

**Note:** For a guide to these key combinations, do either of the following:

- press "Shift-?" on your keyboard
- click the **Help** icon from any page of the EDA GUI and select **Hotkeys**

Where a combination is shown as <key> + <key>, press the second key simultaneously or 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 4: Global shortcuts (available from all pages)*

| Key combination | Action |
|---|---|
| ? | Show hotkey menu |
| g+h | Navigates to **Home** page |
| g+a | Navigates to the **Alarms** page |
| g+q | Navigates to the **Queries** page |
| g+t | Navigates to the **Topologies** page |
| g+r | Navigates to the **Transactions** page |
| g+w | Navigates to the **Workflows** page |

*Table 5: 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. |

| Key | Action |
|---|---|
| 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. |
| space | Toggles the focused (active) check box in the row. |

*Table 6: Form shortcuts*

| Key | Action |
|---|---|
| shift-y | Toggle the YAML/JSON configuration view open or close |
| shift-d | Toggle the descriptions of the properties |
| shift-f | Takes cursor to the field search box |
| shift-t | Add form to transaction |
| shift-c | Commit the form |

## 3.7 Working with the EDA menu

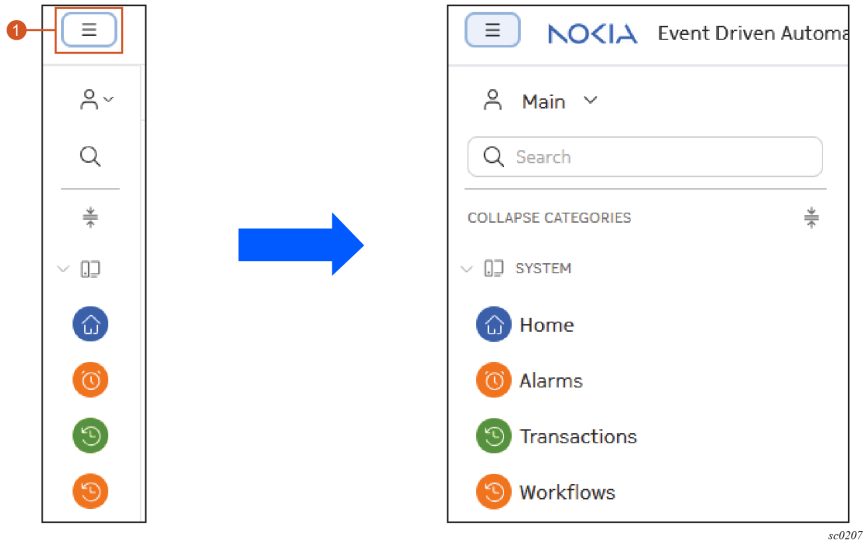
The EDA menu always includes a set of basic selections. As you install new apps, the menu grows to include the new selections provided by those apps.

Within the EDA GUI there are some options to make it easier for you to access, navigate through, and find things within the EDA main menu.

### Pinning and un-pinning the main menu

Click the hamburger button at the upper left off the EDA GUI to expand the menu horizontally and pin it open, or to un-pin it and allow it to collapse against the left edge of the EDA GUI.

*Figure 10: The EDA main menu showing an expanded category*



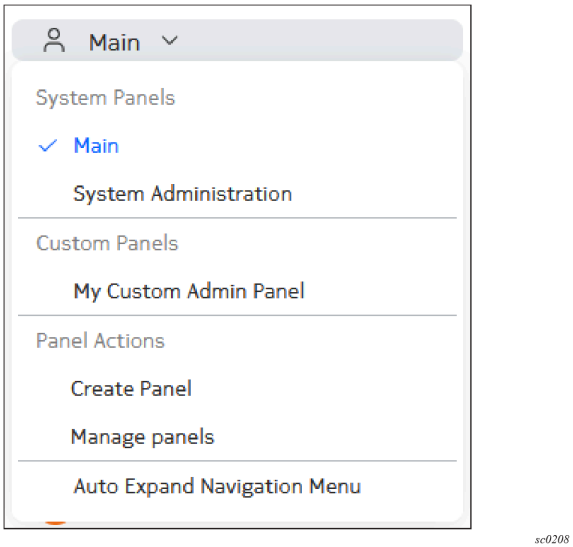

## Major menu groups

The EDA menu is divided into two major groups, or panels:

- **Main**: this panel contains the majority of options you can use to manage resources with EDA.
- **System Administration**: this panel contains options you can use to manage the EDA application itself, as well as user management and node management selections.

You can use the menu options to display only the **Main** menu; or only the **System Administration** menu; or both (**All**).

*Figure 11: Menu display options*



## Custom menu panels

To help you navigate efficiently to pages within the EDA GUI, you can create and share your own custom menu panels that contain and organize links to pages of most interest to you.

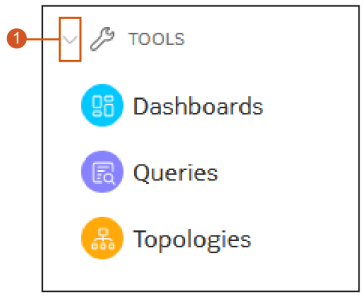
## Menu categories

The EDA menu groups related menu selections together into a set of categories.

- Menu categories appear in gray text and have an associated icon.
- Applications and resources appear in black text and have an associated circle containing an icon or letters.

You can vertically expand or collapse individual categories using the chevron beside the category title.

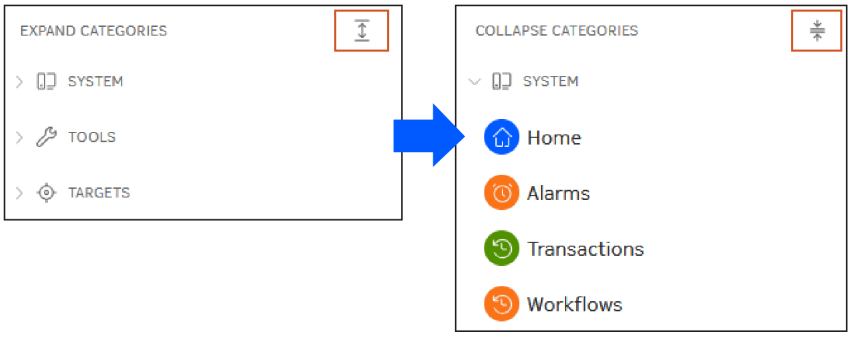
*Figure 12: Menu expand/collapse chevron*




You can also expand or collapse all categories simultaneously using the **EXPAND CATEGORIES** or **COLLAPSE CATEGORIES** toggle near the top of the menu.

When collapsed, the categories display only their icons. When expanded, the categories display their label text as well.

*Figure 13: Expand menu toggle*

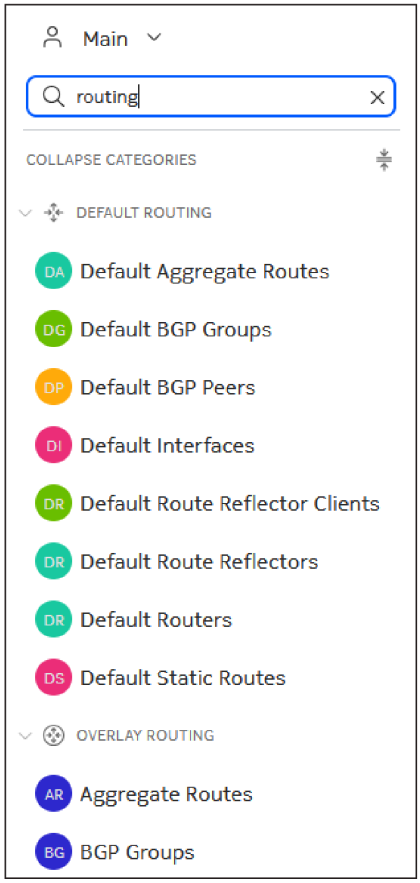



## Searching the menu

You can use the Search box to find all of the menu selections whose names include a particular string of text; the menu displays only matching options.

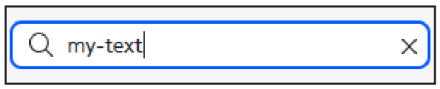
If your search matches a category name, all of the selections within that category are included in the search results (even though the selections themselves may not match your search string).

*Figure 14: Menu search results*




sc0211

To clear the text within the Search field, click the **X** at the right of the field.




sc0250

**Related topics**

*Custom navigation panels*

## 3.8 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:** Some lists include special options uniquely available for lists of particular data. Those options are described in the topics pertaining to those lists.

*Figure 15: A sample data grid with controls*

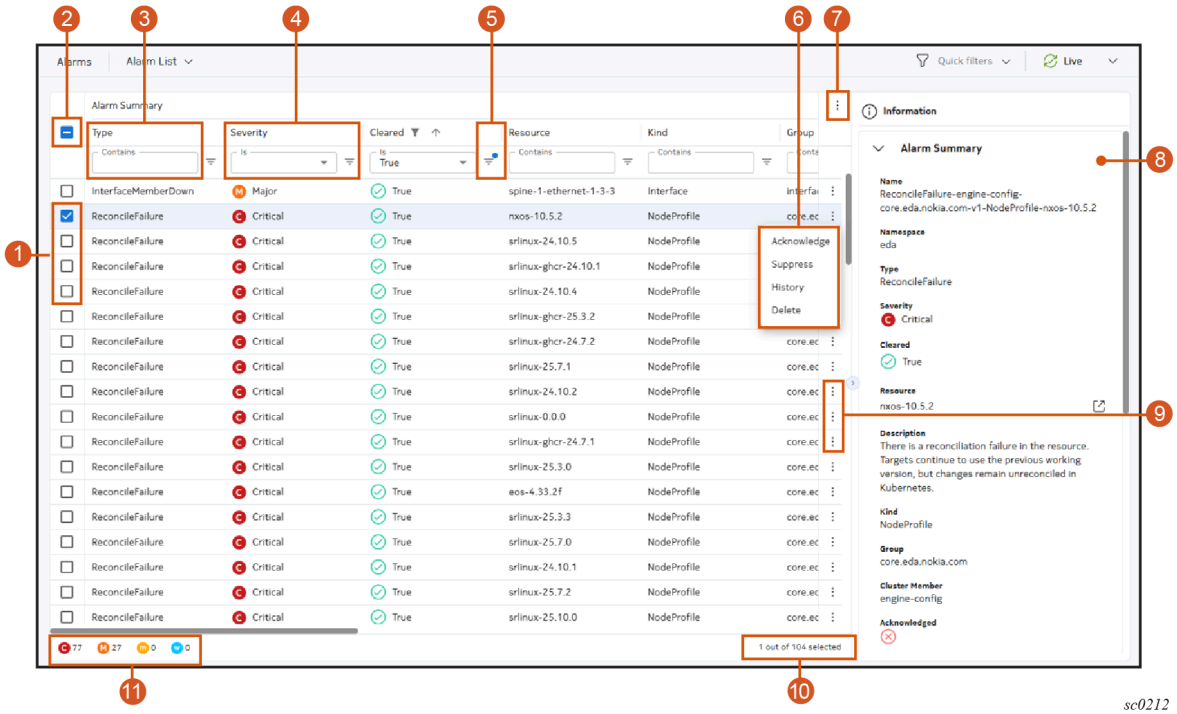



*Table 7: Data grid controls*

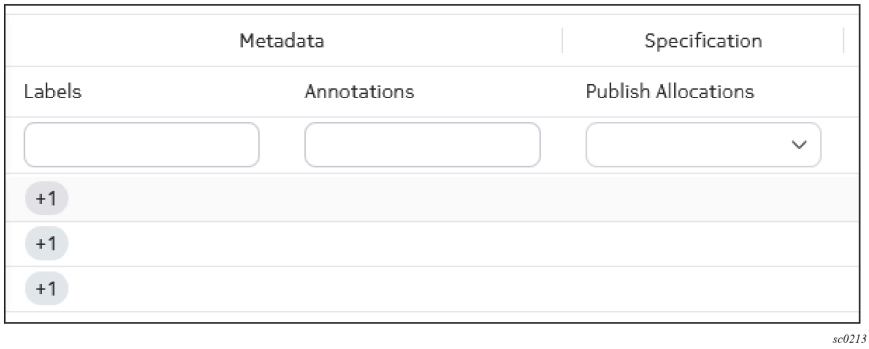
| # | Name | Function |
|---|---|---|
| 1 | Row selectors | Use these to select one or more individual rows.<br>**Note:** You can use the space key to toggle the focused (active) check box in the row. |
| 2 | Multiple row selector | Use this to toggle between selecting all rows, and un-selecting all rows. |
| 3 | Text string filter | Enter an alphanumeric string to filter the list based on matching values in that column. |
| 4 | Selection filter | Select a value to filter the list based on matching values in that column. |
| 5 | Filter Applied indicator | When a dot is superimposed on the Filter icon, this field is currently applying a filter to the displayed list. |
| 6 | Table settings and actions button | Click to open the menu of standard table actions, as well as the list of multi-row actions available for this table. |

| # | Name | Function |
|---|---|---|
| 6 | Table row actions | An example of the actions menu that is displayed after clicking the **Table row actions** button. |
| 7 | Table settings and actions button | Click to open the menu of standard table actions, as well as the list of multi-row actions available for this table. |
| 8 | Information panel | Like other pages in EDA, this panel displays detailed information about the selected object – in this case, the row or rows that are selected in the data grid. |
| 9 | Table row actions button | Click to select from a set of actions specific to the data in that row. |
| 10 | Row and selection counter | Indicates the total number of rows, and the number of rows that are currently selected. |
| 11 | Category counter | The Alarms data grid shown as an example includes an indicator of how many alarms in the full list belong to each severity category. |

### 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 16: Nested columns for metadata*




### 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 rearrange the position of a column, click the column header and drag it to its new position.

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.

For most data grids, the **Name** and **Namespace** columns are pinned to the left by default. When scrolling horizontally, these columns remain visible at the left edge of the display.

### Dynamic display for namespace column

Many data grids include a column for Namespace, which identifies the particular namespace associated with each row.

However, if you have selected a single namespace in the **Namespace** selector on the top bar, then the Namespace column is automatically hidden on the data grid. Because all namespace values would match the selected namespace, displaying the column is not useful.

### Pinning columns

You can pin one or more columns to the left or right side of a data grid. Pinned columns continue to display even as the rest of the data grid scrolls to the left or right.

The set of pinned columns are bounded by a vertical gray line on their right edge.

In most data grids, the **Name** column is pinned by default. However, the default pinned columns can vary by page.

To pin additional columns, click any column header and drag it into the pinned area. To un-pin a column, click the column header and drag it out of the pinned area.

### 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 click 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.

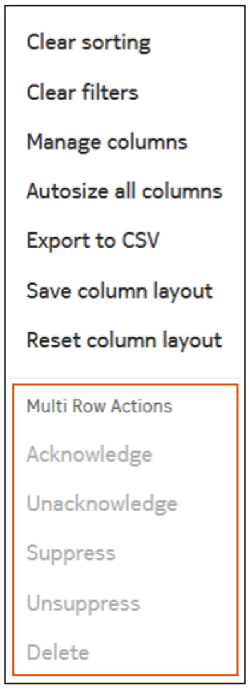
To clear all filtering from the data grid, click the **Table settings & actions** icon and select **Clear filters** from the action list.

### Multi-row actions

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

Some tables also support the option to **Revert** multiple deleted resources simultaneously. For more information on the **Revert** button, see Transactions.

*Figure 17: Multi-row actions for nodes list*

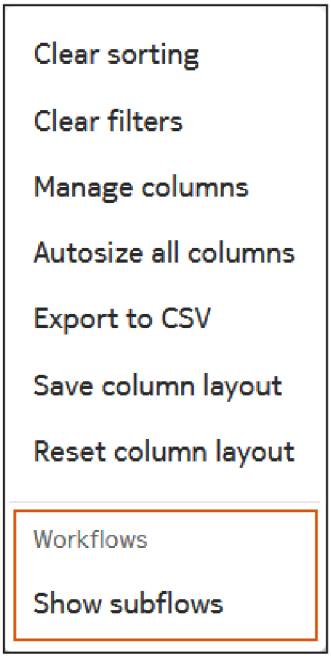



sc0214

### Special actions

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

*Figure 18: Special actions for Workflows*




*sc0215*

### Counters

Many data grids throughout the EDA UI contain counters for alarms and deviations. Each of these counters appears as a column in the data grid.

The Alarm column displays the number of alarms present in each of the following categories for the specified resource:

- critical
- major
- minor
- warning

The Deviation column displays the number of deviations associated with the specified resource.

For Node resources, the alarm counters include both alarms raised against the resource and alarms where the node is listed as an affected target.

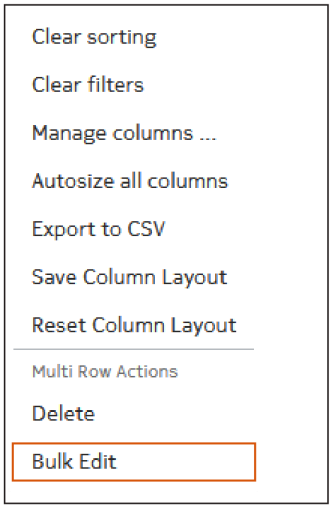
**Related topics**

*Key bindings*
*Namespaces*

## 3.8.1 Bulk edits

To aid on those occasions where you need to make the same change to multiple items, some data grids include a **Bulk Edit** option.

*Figure 19: Actions for the nodes list*




sc0216

The Bulk Edit option allows you to:

1. Select a set of objects in a list.
2. Configure a set of properties they all share in common to be removed, added, or replaced.
3. Apply those same changes to all of the selected objects (immediately as a commit, or later as part of a transaction).

After you select all of the objects that are the subject of your change (using the check box at the left of each row), selecting the **Bulk Edit** option opens the **Bulk Edit** page.

The **Bulk Edit** page indicates the number of objects selected, and lists all of the editable properties those objects share. For each property, a check box allows you to select it for modification.

After an item has been selected for editing, a drop-down control allows you to choose the type of change to make, and a field displays the specific changes you have indicated for that property.

*Figure 20: The bulk edits page*

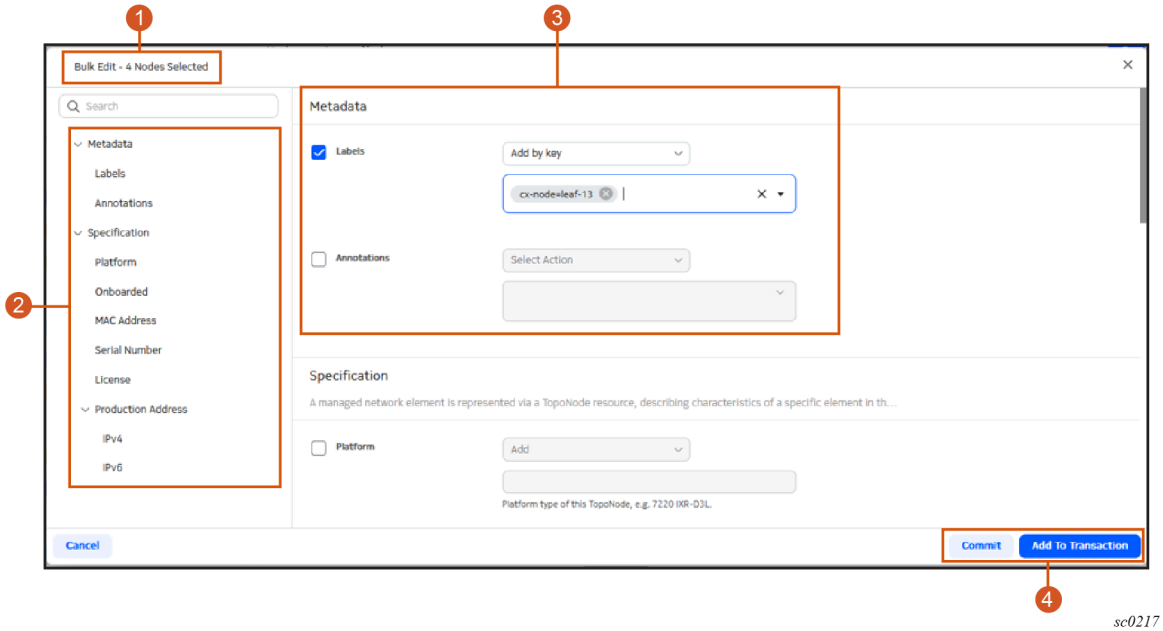



*Table 8: Elements of the Bulk Edit page*

| Item | Description |
|---|---|
| 1 | The page name and an indication of the number of selected objects that are subject to these bulk changes after they are committed. |
| 2 | A list of modifiable properties for the selected objects. |
| 3 | In this case the Labels field is selected for modification. Because the field has been selected for modification, the actions drop-down list is displayed. Available options are specific to the Labels field. |
| 4 | After configuring the set of changes for all parameters, choose from among the standard Commit options for this bulk edit:<br>• **Commit** to immediately apply the changes on this **Bulk Edit** page.<br>• **Add to Transaction** to store these changes to be processed later as part of a transaction (which can include other accumulated commits to be applied as part of the same operation). |

The actions available for a specific parameter as part of a bulk edit depend on the type of data being modified.

## Bulk edits for single-value fields

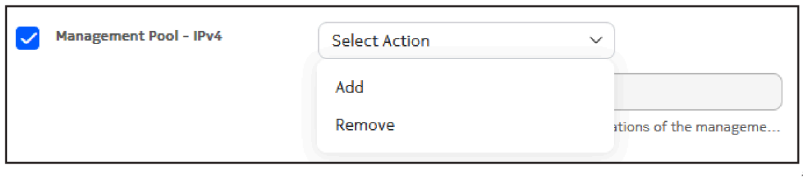
Single-value fields are those that contain integers, strings, or enums (single selections from a drop-down list).

- **Add**: the new value supplied as part of the edit is written to the selected field.

- **Remove**: any current values in the selected field are deleted.

**Note:** If a field is required, the Remove option is not available.

*Figure 21: Bulk edit options for an optional text field*




## Bulk edits for maps

Maps type fields are collections of key-value pairs. Supported bulk edit options can be based on the key, or on the field as a whole.

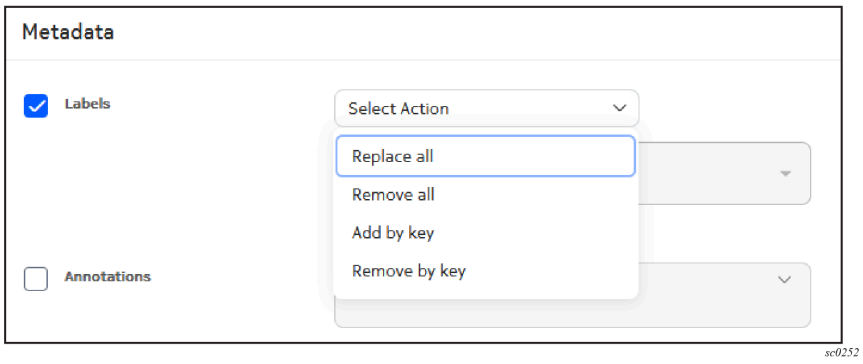
- **Replace all** overwrites the current set of objects with the values supplied as part of the edit.
- **Remove all** removes all objects in the array. This is only available for optional arrays.
- **Add by key** appends new values supplied as part of the edit to the current set. If a key used in the new values matches that of an existing key-value pair, the existing value is overwritten with the new value.

## Bulk edits for labels

Labels are a special case of maps with support for "Remove by key" in addition to the other map actions.

- **Replace all** overwrites the current set of objects with the values supplied as part of the edit.
- **Remove all** removes all objects in the array. This is only available for optional arrays.
- **Add by key** appends new values supplied as part of the edit to the current set. If a key used in the new values matches that of an existing key-value pair, the existing value is overwritten with the new value.
- **Remove by key** removes objects from the current set that match the specified key.

*Figure 22: Bulk edit options for a label field*



## 3.9 Common data grid 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.

**Note:**
In addition to the common actions described above, the list of actions may also include:

- Multi-row actions that can be applied to multiple selected items simultaneously.
- Special actions that are unique to specific types of resources.
- Bulk edits, which allow you to modify multiple selected items in the same way.

# 4 Custom navigation panels

The EDA navigation menu provides many links to the tools and resources within the EDA GUI, sorted into categories. These categories and links are organized within separate navigation panels (for example, the standard navigation panels **Main** and **System Administration**).

To limit the menu links to the links you use most often, EDA allows you to create navigation panels containing only the categories and navigation links that you select. You can even create multiple custom navigation panels, so that you can switch between the panels and their menus that are optimized for particular tasks.

You can also include custom links to internal pages, and links to saved dashboards as part of your navigation panel.

To further help you manage navigation, you can also show or hide any of the standard or custom EDA navigation panels.

Using the **Manage Panels** option in the EDA main menu, you can:

- Build a custom navigation panel that includes the links to the pages you use most, organized into categories that work for you
- Edit or duplicate a custom panel (a duplicated navigation panel generally serves as the starting point for creating a modified version of it)
- Hide or delete a custom panel
- Share your custom panel with other EDA users

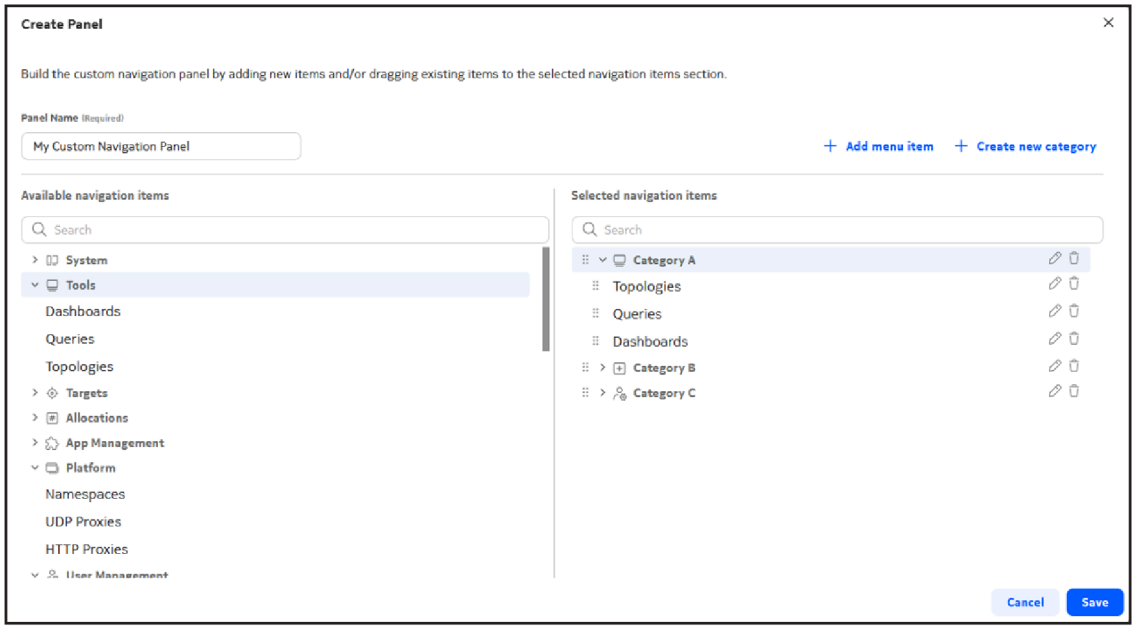
**Note:** Published panels are visible to all EDA users with read permissions for shared user storage.

## 4.1 Custom Panel page

The **Custom Panel** page is used, with a minor change in title, whenever you:

- Create a new custom navigation panel
- Edit an existing custom navigation panel
- Open a duplicate of an existing navigation panel, to modify it and save it as a new panel

*Figure 23: The custom navigation panel creation form*




*sc0271*

*Table 9: Elements of the Custom Navigation Panel Creation form*

| Dashlet | Description |
|---|---|
| Panel name | Enter the name to be displayed in the EDA navigation menu for this panel. |
| Add menu items link | Opens a form on which to define a custom menu item and add it to an existing category within the navigation panel you are designing. |
| Create new category link | Opens a form on which to define a custom menu category with a name and associated icon, and add it to an existing category within the navigation panel you are designing. |
| Available navigation items | A full list of standard EDA menu categories and their available navigation items. Drag whole categories or individual navigation items to any position within the navigation panel you are designing. |
| Selected navigation items | A preview of the navigation panel you are designing. |

## 4.2 Creating a custom panel

### About this task

A custom navigation panel is an EDA navigation panel that you construct yourself, containing menu categories and links to pages within the EDA GUI.

After you construct a custom navigation panel, it appears in the list of selectable navigation panels in the EDA main menu. You can then select the custom navigation panel to display it instead of, or in addition to, the standard EDA main menu.

Optionally, instead of constructing the entire navigation panel from scratch, you can duplicate an existing navigation panel and then add and remove categories and menu items as required.

### Procedure

**Step 1.** Do either of the following:

- To build a custom navigation panel from scratch, use the **Main** navigation panel to select **Create Panel**.
- To start your design from an existing navigation panel, use the **Main** navigation panel to select **Manage Panels**. Select a navigation panel from the resulting list, and click **Duplicate Panel** from the available actions for that selection.

**Expected outcome**

The **Create Panel** or **Duplicate Panel** displays, depending on your selection.

**Step 2.** Enter a name for your panel in the **Panel Name** field.

**Step 3.** Add one or more categories for your navigation panel.

**a.** Click **+ Create new category** to open the **Create Category** form.

**b.** Enter a name in the **Category Name** field.

**c.** Select an icon to display next to this category in the custom navigation panel.

**d.** Repeat 3.a to 3.c until you have added all of the required categories to your navigation panel design.

**e.** Click **Save**.

**Step 4.** Optionally, add one or more custom menu items to your navigation panel and define their target.

**a.** Click **+ Add menu item** to open the **Create Custom Menu** form.

**b.** Select a **Category** from within your custom navigation panel to contain your menu item.

**c.** Enter a **Custom Menu Name** for your menu item.

**d.** Specify the target page for the custom link by providing a URL in the **Custom Menu Link** field. Both internal and external links are supported.

- Internal links for the Topologies page within the EDA GUI are defined by their relative path. For example: `/ui/app/main/topologies.eda.nokia.com/v1alpha1/topologies`.
- External links can be defined with a URL. For example: https://docs.eda.dev/.

**e.** If the required **Custom Menu Link** field is set to /ui/main/queryapi, use the **Optional Navigation Query** field to define an EQL query to load when opening the target page.

**Note:** The **Optional Navigation Query** field only functions if you select the `/ui/main/queryapi` endpoint in the navigation field as described above. No other menu options can incorporate this query into their navigation.

**f.** Use the **Open link in new tab** check box to indicate whether the target page should replace the current tab, or open a new tab when it opens.

**Note:** External links always open in a new tab.

**g.** Repeat 4.a through 4.f until you have added all of the required custom menu items to your navigation panel design.

**h.** Click **Save**.

**Expected outcome**

Your custom menu item is added to the navigation panel design, under the category you selected.

**Step 5.** Add one or more standard categories or individual menu items from the EDA main menu to your navigation panel design.

**a.** Do either of the following to find your intended category or menu item:

- Expand any of the standard menu categories that are listed under **Available menu items** to reveal the intended menu item.
- Use the Search field under **Available menu items** to find a particular category or menu item.

**b.** Click and drag the category or menu item from the **Available menu items** to the intended position within your navigation panel design.

**Note:** If you drag a menu category from **Available menu items**, all of its menu items are also added to your panel design under that category.

**c.** Repeat 5.a and 5.b until you have added all of the required categories and menu items to your navigation panel design.

**Step 6.** To remove a category or menu item from your navigation panel design by clicking the trash icon next to that item in the **Selected menu items** list.

**Note:** This action is most useful if you began by duplicating another, complete menu from which you would like to trim some items.

**Step 7.** To edit a category or menu item in your navigation panel design, do the following:

**a.** Click the pencil icon next to that item in the **Selected menu items** list.

**b.** In the resulting Edit form, do any of the following:

- For a custom or standard category: edit the category name or change the associated icon.
- For a custom or standard menu item: change any of the properties of the menu item, including:
  - Category
  - Custom Menu Name
  - Custom menu Link
  - Optional Navigation Query
  - Open link in new tab check box value

You can also reorder categories and the custom menus inside them, as well as move the custom menus between categories in the panel you are building.

**Step 8.** Click **Save**.

## 4.3 Managing custom panels

### About this task

EDA supports the following actions for custom navigation panels:

- Duplicate an existing standard or custom navigation panel
- Hide an existing standard or custom navigation panel
- Re-display any hidden navigation panel
- Edit an existing custom navigation panel
- Delete an existing custom navigation panel
- Publish a custom panel to share it with other EDA users

**Note:** Sharing a panel automatically makes it available to all users who have access to the shared folder. The ability to see or edit shared panels can be managed through security permissions applied on the ClusterRole URL rule for the path `/core/user-storage/v2/shared/file`.

**Note:** You cannot edit or delete the standard Main or System Administration navigation panels.

### Procedure

**Step 1.** Use the **Main** navigation panel to select **Manage Panels** to open the **Manage Panels** form.

**Step 2.** Find the navigation panel you want to work with in the resulting list.

**Step 3.** Select the action that you want from the **Table row action** drop-down list:

- To duplicate a panel, go to step 4.
- To hide a navigation panel, click **Hide Panel** and go to step 7.
- To re-display a hidden navigation panel, click **Unhide Panel** and go to step 7.
- To edit a navigation panel, go to step 5.
- To publish (share) a navigation panel, click **Publish** and go to step 7.
- To delete a navigation panel, go to step 6.

**Step 4.** To duplicate the navigation panel, do the following:

**a.** Click **Duplicate Panel** to open the **Duplicate Panel** page.

**b.** Name and modify the duplicate panel, following the procedure for managing content for a new custom navigation panel.

**c.** Click **Save**.

**d.** Go to step 7.

**Step 5.** To edit the selected custom navigation panel, do the following:

**a.** Click **Edit Panel** to open the **Edit Panel** page.

**b.** Modify the panel, following the procedure for managing content for a new custom navigation panel.

**Note:** You cannot edit the name of the custom navigation panel.

**c.** Click **Save**.

**d.** Go to step 7.

**Step 6.** To delete the selected custom navigation panel, do the following:

**a.** Click **Delete Panel**.

**b.** Click **OK** in the confirmation message.

**Step 7.** Click the **Close** button to close the form and complete this procedure.

# 5 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 is displayed, set thresholds for highlighting, and make other formatting choices depending on the dashlet type.

Dashboard builder also supports filters. Filters use variable substitution to modify dashlet queries based on user input. For example, a dashboard which displays data for all TopoNodes can be filtered to display only data from a specific TopoNode.

The Dashboard designer displays dashboards that can originate from three sources (as indicated in the **Source** column):

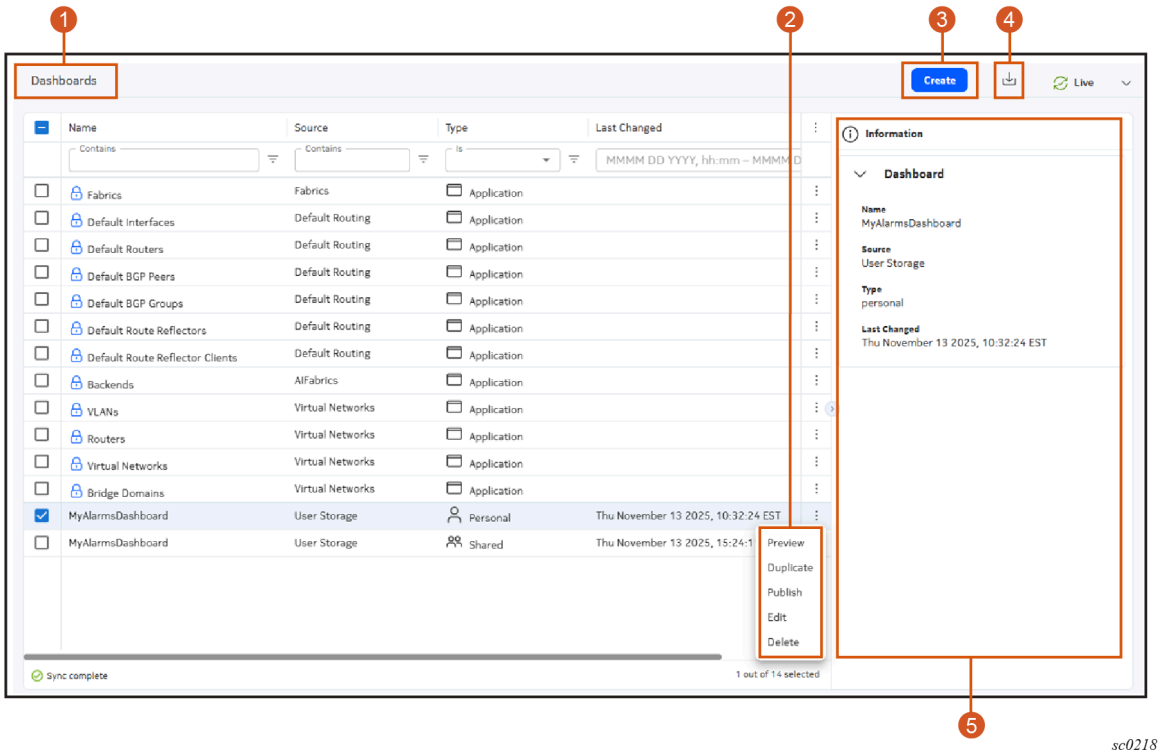
- dashboards that were created by EDA and its installed applications
- dashboards created by the current user
- dashboards that the current or other users have chosen to share.

## 5.1 Dashboards page

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

The Dashboards list displays all of the dashboards available to you. Open the Dashboards list by selecting **Dashboards** from the main navigation panel.

*Figure 24: The dashboards page*




## Elements of the Dashboard list

*Table 10: Elements of the Dashboards page*

| # | Name | Function |
|---|---|---|
| 1 | Breadcrumb bar | Displays the current position in the set of Dashboard designer pages. |
| 2 | Row actions | Clicking the Table row actions icon reveals the actions available for the current row in the dashboards data grid.<br>See Row actions for dashboards. |
| 3 | **Create** button | Click to open the Dashboard Designer view for a new dashboard. |
| 5 | **Import Layout** button | 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. |
| 6 | Information panel | A standard EDA information panel, displaying details about the dashboard that is currently selected in the dashboard list. |

*Table 11: Default Dashboard display columns*

| Column | Description |
|---|---|
| Name | The display name configured for the dashboard. |
| Source | The source from which the dashboard originated. This can be one of the following values:<br>• An app name (such as Fabrics or Default Routing): this is the source application for this dashboard.<br>• User Storage: this is a user-created dashboard. |
| Type | • Application: a dashboard associated with EDA or one of its installed apps. You cannot edit, delete, or publish these dashboards, as signified by the lock icon beside the dashboard name.<br>• Personal: a dashboard created by the current user account.<br>• Shared: a user-created dashboard that has been published for sharing. When you share your own dashboards, the Shared version is a different entity than the original Personal dashboard. Both can be edited independently. |
| Last Changed | The time and date that the last modification was saved for this dashboard. |

*Table 12: Non-default Dashboard display columns*

| Column | Description |
|---|---|
| ID | A unique ID assigned internally by the EDA system. |
| File Name | The name and path for the JSON file that contains this dashboard configuration. For example, services/ui/router-dashboard-v0.2.json. |
| Group | The group within the EDA database where this dashboard is stored. For example, services.eda.nokia.com |
| Version | The current version of the dashboard. For example, v1.<br>Editing and saving a dashboard creates a new version. |

### Row actions for dashboards

The following row actions are available for dashboards in the list:

*Table 13:*

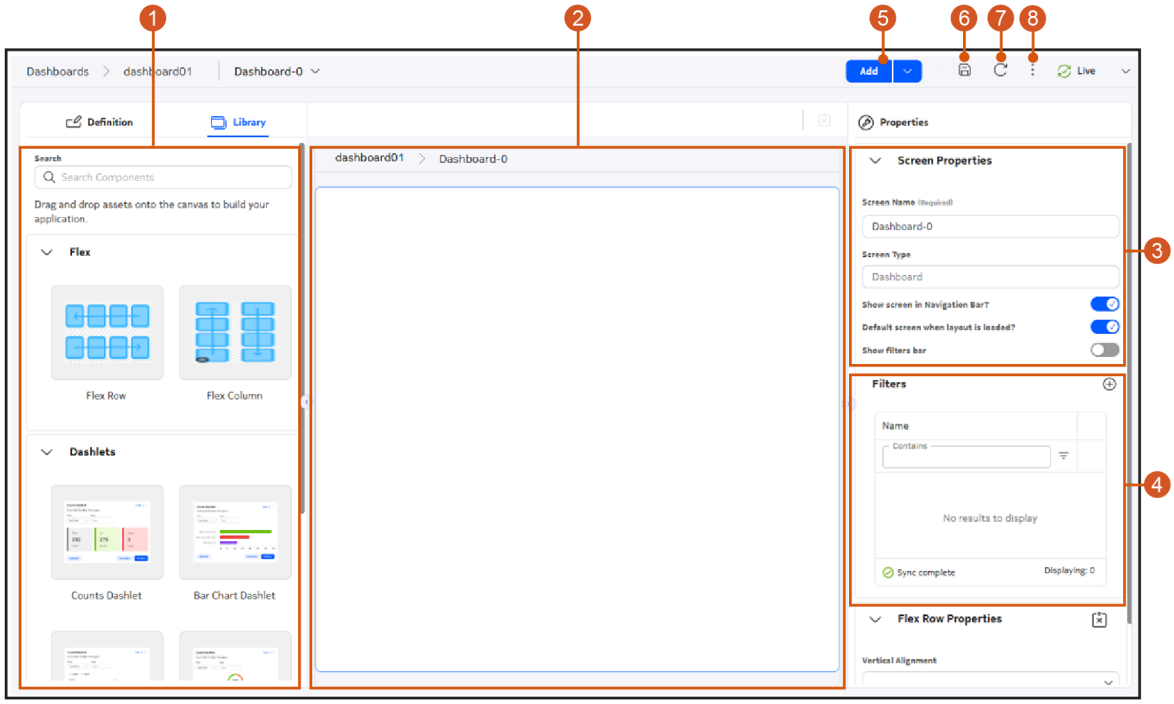
| Action | Description | Application dashboards? | Personal dashboards? | Shared dashboards? |
|---|---|---|---|---|
| Preview | Shows a preview of the dashboard and the data it contains. | Yes | Yes | Yes |

| Action | Description | Application dashboards? | Personal dashboards? | Shared dashboards? |
|---|---|---|---|---|
| Duplicate | Make a copy of the current dashboard, which you can save under a different name and then modify as required. | Yes | Yes | Yes |
| Publish | Publish a Personal dashboard for sharing with others.<br>If you publish an already-published dashboard, a confirmation dialog warns you that proceeding will over-write the current published version. | No | Yes | No |
| Edit | Open the current dashboard for editing. Editing a Shared dashboard edits only the shared copy, not the personal copy on which it is based. | No | Yes | Yes |
| Delete | Delete the selected dashboard.<br>Delete is also available as a multi-row action, in which case it deletes all of the dashboards currently selected in the list. | No | Yes | Yes |

## 5.2 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 25: The dashboard designer page*




sc0219

*Table 14: Elements of the Dashboard Designer page*

| # | Name | Function |
|---|---|---|
| 1 | Definition/Library panel | The Definition tab displays basic parameters about the current dashboard layout, like its name and description.<br>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.<br>Select objects in this panel to view and configure their properties in the Properties tab. |
| 3 | Properties | 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 basic display properties for the dashboard. |
| 4 | Filter configuration panel | Filters are an optional way to modify the data underlying dashlets contained on the dashboard.<br>Use this panel to configure one or more filters for the current dashboard. |

| # | Name | Function |
|---|---|---|
| | | When a filter is configured, you can include a corresponding "where" clause in the queries underlying individual dashlets.<br>Enable the **Show filters bar** property to display a widget on the layout panel that allows you display and use specific filters you have configured.<br>See the procedure for creating dashboard filters for the steps to create a filter, add a reference to the queries for dashlets, and use the filters to constrain the data displayed by those dashlets. |
| 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:<br>• **Preview saved changes**: open a new tab that displays the current dashboard design.<br>• **Export**: save the dashboard design as a file, which others can import into their copy of EDA. |

## 5.3 Flex grids

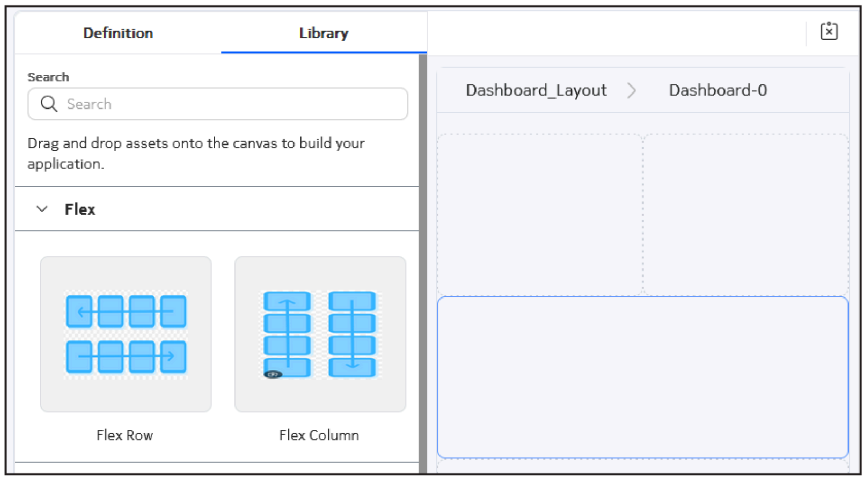
Flex grids are the underlying structure of a dashboard. You must add at least one flex row or flex column to the dashboard before you can then fill the resulting grid with individual dashlets.

To add a flex row or flex column to a dashboard design, drag the flex row or flex column from the Library panel onto the Layout panel.

You can add a series of rows and columns to create a customized grid for your dashboard design. Each cell in the grid can then accept one or more dashlets.

For example, in the illustration below, the dashboard includes multiple flex rows, and a flex column has been added to the first row to divide it into two cells.

*Figure 26: Flex grid*




When you select a specific cell in the Layout panel, the Properties panel displays two properties that you can configure for the selected flex row or column:

- **Vertical Alignment**: the vertical position of any dashlets that are placed in that cell.
- **Horizontal Alignment**: the horizontal position of any dashlets that are placed in that cell.

## 5.4 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.

### 5.4.1 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 27: A sample counter dashlet*

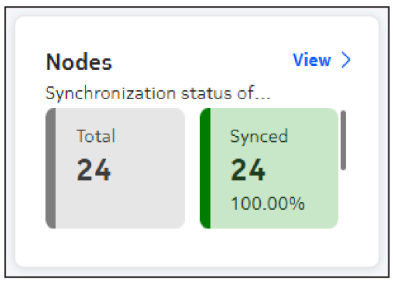


*Table 15: Count dashlet properties*

| 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, when clicked, opens a selected page within the EDA UI. The target can be:<br>• a page within the EDA GUI that you select from the displayed drop-down list<br>• the Query Builder page. For this option, enter the navigation target / ui/main/queryapi. This causes an additional field to display among the dashlet properties: **Optional Navigation Query**.<br>The query that is displayed on the Query Builder page after a user clicks the **View** link on the dashlet depends on the value in the **Optional Navigation Query** field. Here you can set the query that displays in the Query Builder; this could be the query that underlies the dashlet, or another. |
| Fill available width | Dynamically changes the dashlet width based on the browser window and neighboring dashlets. |
| Dashlet width | The relative width of the dashlet. |
| Dashlet height | The relative 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.<br>On that page you configure the data source as one of the following:<br>**1.** EQL Query<br>**2.** Natural Language query<br>**3.** GVK Definition<br>**4.** URL Endpoint |
| Counters | These properties configure the highlighting of values that meet criteria on the counts dashlet:<br>• **Label**: the label shown beside qualifying values<br>• **Color**: the color used to highlight qualifying values<br>• **Field**: the field within the data source to be evaluated for possible highlighting<br>• **Criteria** (Equals, Not Equal, Greater Than, Less Than): the logical operator that qualifies for this highlight (in combination with Value)<br>• **Value**: the comparison value for the logical criterion. |

| Property | Description |
|---|---|
| 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. Changes 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. Changes between the total being above, or below the count of qualifying values. |

### 5.4.2 The line chart dashlet

A line chart dashlet places a line chart on the dashboard layout. 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 28: A sample line chart dashlet*

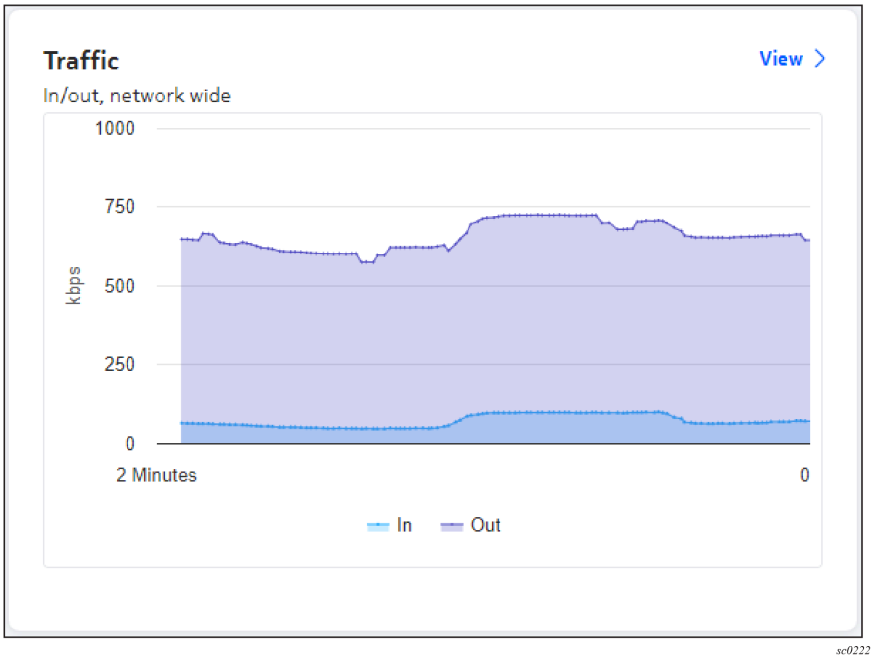



*Table 16: Line chart dashlet properties*

| Property | Description |
|---|---|
| Common properties | |

| Property | Description |
|---|---|
| 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, when clicked, opens a selected page within the EDA UI. The target can be:<br>• a page within the EDA GUI that you select from the displayed drop-down list<br>• the Query Builder page. For this option, enter the navigation target / ui/main/queryapi. This causes an additional field to display among the dashlet properties: **Optional Navigation Query**.<br>The query that is displayed on the Query Builder page after a user clicks the **View** link on the dashlet depends on the value in the **Optional Navigation Query** field. Here you can set the query that displays in the Query Builder; this could be the query that underlies the dashlet, or another. |
| Fill available width | Dynamically changes the dashlet width based on the browser window and neighboring dashlets. |
| Dashlet width | The relative width of the dashlet. |
| Dashlet height | The relative 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.<br>On that page you configure the data source as one of the following:<br>**1.** EQL Query<br>**2.** Natural Language query<br>**3.** GVK Definition<br>**4.** URL Endpoint |
| Chart Configuration | These properties control the display of the line chart:<br>• Maximum number of data points<br>• Y-Axis Units<br>• Scaling Function (None, Metric Prefix Scaling) |

### 5.4.3 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 29: A sample donut dashlet*

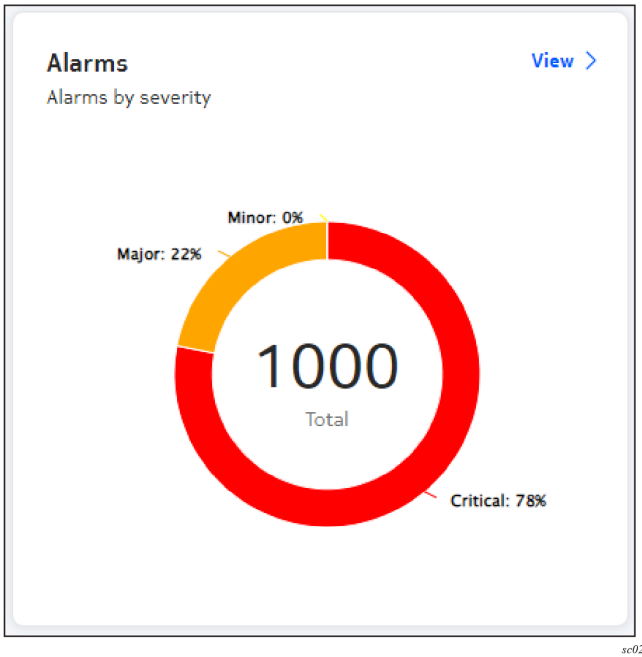



*Table 17: Donut dashlet properties*

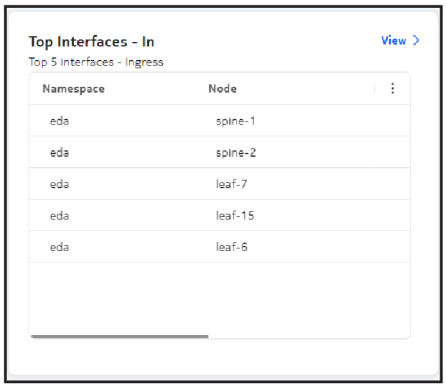
| Property | Description |
|---|---|
| Common properties | |
| Title | The title of the dashlet when displayed in the EDA UI. |
| Subtitle | A subtitle, displayed below the title and in a smaller font. |
| Navigation target | Adds a **View** button to the dashlet that, when clicked, opens a selected page within the EDA UI. The target can be:<br>• a page within the EDA GUI that you select from the displayed drop-down list<br>• the Query Builder page. For this option, enter the navigation target / ui/main/queryapi. This causes an additional field to display among the dashlet properties: **Optional Navigation Query**.<br>The query that is displayed on the Query Builder page after a user clicks the **View** link on the dashlet depends on the value in the **Optional Navigation Query** field. Here you can set the query that displays in the Query Builder; this could be the query that underlies the dashlet, or another. |
| Fill available width | Dynamically changes the dashlet width based on the browser window and neighboring dashlets. |
| Dashlet width | The relative width of the dashlet. |

| Property | Description |
|---|---|
| Dashlet height | The relative height of the dashlet. |
| Charts | |
| Charts | A single pie chart dashlet can include multiple pie charts.<br>Use this space to add and configure each pie chart.<br>After configuring a pie chart, click the + icon to add and configure an additional pie chart for this dashlet. |
| Donut Chart Details<br>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.<br>Choose from:<br>• EQL Query<br>• Natural Language query<br>• GVK Definition<br>• URL Endpoint |
| Hide title | Indicates whether to show the chart title on the chart, or not.<br>Options: Yes or No. |
| Show total | Indicates whether the sum of all segments should be displayed on the chart, or not.<br>Options: Yes or No |
| Show slice labels | Indicates whether each chart segment should display a label for its data.<br>Possible values: All, Percent, None |
| Segments: these properties control the display of each segment in the chart.<br>Configure and add as many segments 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 that corresponds 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. |

### 5.4.4 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 30: A sample dataview dashlet*




*sc0262*

*Table 18: 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, when clicked, opens a selected page within the EDA UI. The target can be:<br>• a page within the EDA GUI that you select from the displayed drop-down list<br>• the Query Builder page. For this option, enter the navigation target / ui/main/queryapi. This causes an additional field to display among the dashlet properties: **Optional Navigation Query**.<br>The query that is displayed on the Query Builder page after a user clicks the **View** link on the dashlet depends on the value in the **Optional Navigation Query** field. Here you can set the query that displays in the Query Builder; this could be the query that underlies the dashlet, or another. |
| Fill available width | Dynamically changes the dashlet width based on the browser window and neighboring dashlets. |
| Dashlet width | The relative width of the dashlet. |
| Dashlet height | The relative height of the dashlet. |

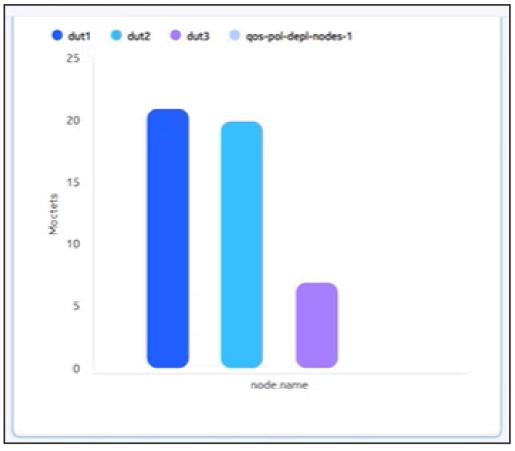
| Property | Description |
|---|---|
| Charts | |
| Query | Click the More icon to open a page on which to configure the data source for this dashlet.<br>On that page you configure the data source as one of the following:<br>**1.** EQL Query<br>**2.** Natural Language query<br>**3.** GVK Definition<br>**4.** 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. |

### 5.4.5 The bar chart dashlet

A bar chart dashlet places a bar chart on the dashboard layout. It supports both horizontal and vertical bar charts.

You can also configure the chart to show stacked bars contributing to a total value, with the elements in the stack indicated as either a raw value or a percentage of the whole.

*Figure 31: A sample bar chart dashlet*




sc0224

*Table 19: Bar chart dashlet properties*

| Property | Description |
|---|---|
| Common properties | |

| Property | Description |
|---|---|
| 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, when clicked, opens a selected page within the EDA UI. The target can be:<br>• a page within the EDA GUI that you select from the displayed drop-down list<br>• the Query Builder page. For this option, enter the navigation target / ui/main/queryapi. This causes an additional field to display among the dashlet properties: **Optional Navigation Query**.<br>The query that is displayed on the Query Builder page after a user clicks the **View** link on the dashlet depends on the value in the **Optional Navigation Query** field. Here you can set the query that displays in the Query Builder; this could be the query that underlies the dashlet, or another. |
| Fill available width | Dynamically changes the dashlet width based on the browser window and neighboring dashlets. |
| Dashlet width | The relative width of the dashlet. |
| Dashlet height | The relative 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.<br>On that page you configure the data source as one of the following:<br>**1.** EQL Query<br>**2.** Natural Language query<br>**3.** GVK Definition<br>**4.** URL Endpoint |
| Chart Configuration | These properties control the display of the line chart:<br>• Group By<br>• Secondary Grouping<br>• Value Field<br>• Unit of Measure<br>• Scaling Function (None, Metric Prefix Scaling)<br>• Use Columns instead of Bars (yes/no)<br>• Show stacked data (Off/Value/Percent) |

## 5.5 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.** Use the **Main** navigation panel to select **Dashboards**.

**Step 2.** Click **Create**.

**Step 3.** Choose one of the following:

- To create a dashboard with a list layout, go to step 4.
- To create a dashboard with a set of dashlets, go to step 10.

**Step 4.** Use the drop-down control beside **Add** to click **Add List Layout**.

**Step 5.** 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 6.** Click the list in the center configuration panel to reveal properties for the list in the Properties panel.

**Step 7.** Configure display properties for the list:

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

**Step 8.** Configure the source data for the list:

**a.** Click the vertical dots 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 9.** Go to step 20.

**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?**:

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

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

**Step 12.** Optionally configure one or more filters for this dashboard.

**Note:** See the separate procedure for creating a dashboard filter for the steps to:

- configure one or more dashboard filters
- modify the queries underlying one or more dashlets to incorporate those filters

**Step 13.** Click the **Library** tab to configure the dashboard layout.

**Step 14.** 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 15.** 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 16.** Click the dashlet in the center configuration panel to reveal properties for the dashlet in the Properties panel.

**Step 17.** 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 18.** To configure the source data for the dashlet (among the dashlet properties):

**a.** Click the vertical dots icon beside the **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 19.** Repeat steps 15, 17 and 18 to add more dashlets to the dashboard if required, until all dashlets are configured.

**Step 20.** Do any of the following:

- To save your dashboard, click the **Save Layout** 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.

## 5.6 Creating a dashboard filter

### About this task

Filters allow you to constrain the data used by the dashlets on a dashboard. After you have configured filters, you can modify the underlying query of any dashlet on the dashboard to include a reference to one or more of those filters. When you specify a value within the filter field on the dashboard, those dashlets display only results that satisfy that filter.
For example, a dashboard which displays data for all TopoNodes can be filtered to display only data from a specific TopoNode you specify in the filter field.

At a high level, configuring and using filters involves the following steps:

1. Configure one or more filters on the dashboard design page.
2. For each dashlet to which a dashboard filter should apply, edit the dashlet's underlying query to include a "where" expression referring to one or more of the dashboard filters.

   **Note:** A sample of each filter's "where" expression is displayed on this page to help you with the correct syntax. A field also displays on this page for each dashboard filter, and you can test the filters by entering values in those fields.

3. On the dashboard design page, enter values into one or more of the filter fields. All dashlets on the dashboard whose underlying query incorporates a filter then update to display only results that satisfy that filter.

The dashboard designer supports three kinds of filters:

- String filter: allows you to add a simple string filter field. As part of each dashlet's query design, you specify a field within the data set against which the filter string is compared. The "where" clause for a string filter resembles `where (<column> = "${StringFilter}").`
- Custom filter: allows you to create a specify a particular field among query results, whose value must match what is entered later as the filter value. The "where" clause for a custom filter resembles `where (<column> = "${CustomFilter}").`
- Name/Namespace: constrains query results to only those associated with a particular name and namespace you enter later as the filter value.
  A Name/Namespace filter is like a Custom filter for which Name/Namespace is hard-coded as the selected filter field. The "where" clause for a Name/Namespace filter resembles `where (<column1> = "${NNFilter.name}" and <column2> = "${NNFilter.namespace}")`

For example, the following query includes a where clause referring to three different filters:

```
.namespace.node.srl.interface.statistics where (out-octets != 0 and in-octets != 0
 and .namespace.node.name = "${nodename}" and .namespace.name = "${target.namespace}"
 and .namespace.node.srl.interface.name = "${custom}") delta milliseconds 250
```

After you have configured one or more filters, corresponding filter fields are displayed in the following places in the EDA GUI:

- on the query configuration page for every dashlet within the dashboard. On the query configuration page you can add the filter expression to the dashlet's underlying query. When the query includes the filter expression, you can test the result by entering filter values to immediately constrain the dashlet's underlying data set.

- on the dashboard configuration page (if you enable their display). Entering a filter value here immediately impacts the data displayed in each dashlet whose underlying data set refers to the filter.
- on the dashlet itself.

If a dashlet's Navigation Target is set to the Query Builder page, then filters can also have an effect on the query displayed on that page.

If you have configured filters for the dashboard, the filter widget displays at the top of the dashboard. Clicking the widget allows you to select a filter field; and you can then enter a value in that field to be applied to the query.

The query that is displayed on the Query Builder page includes the "where" clause but its content depends on whether you have applied any filters:

- if you have applied no filters, the "where" clause is present, but all filter values are set to wildcards (*) so the filter has no effect.
- if you have applied one or more filters, the filter "where" clause is present, and the filter values you entered are included within the query string and are applied to the displayed results.

### Procedure

**Step 1.** Click the + beside the **Filters** label.

**Step 2.** Select the type of filter you are creating in the **Type** list control:

- to add a string filter: go to step 3.
- to add a Name/Namespace filter: go to step 5.
- to add a custom filter: go to step 7.

**Step 3.** Configure a String filter for the dashboard by doing the following:

**a.** Enter a **Name** for the filter.

**b.** Click **Save**.

**Step 4.** Go to step 8.

**Step 5.** Configure a Name/Namespace filter for the dashboard by doing the following:

**a.** Enter a **Name** for the filter.

**b.** Use the **Query** field to create a query that generates the data set against which this filter is applied. This query can be in the form of an EQL Query, Natural Language query, GVK Definition, or URL Endpoint.

**Note:** The resulting data set must include the Name and Namespace fields. If those fields are not present in the data set, the filter fails.

**c.** While the focus is still on the **Query** field, press the **Enter** key to signal that you have finished configuring the query.

**Note:** If you do not press the Enter key, the EDA UI does not recognize that the query is complete and you are unable to proceed.

**d.** Click **Save**.

**Step 6.** Go to step 8.

**Step 7.** Configure a Custom filter for the dashboard by doing the following:

a. Enter a **Name** for the filter.

b. Use the **Query** field to create a query that produces the data against which this filter is applied. This query can be in the form of an EQL Query, Natural Language query, GVK Definition, or URL Endpoint.

c. While the focus is still on the **Query** field, press the **Enter** key to signal that you have finished configuring the query.

**Note:** If you do not press the Enter key, the EDA UI does not recognize that the query is complete and you are unable to proceed.

**Note:** You may need to wait a moment for the system to finish resolving the query before selecting a field from the query results in the next step.

d. After you have created the query, and the system has had a chance to resolve it and can identify the fields of data returned, use the **Field** list to select one of the fields within the resulting data set. The field you select here is the field against which the filter text you enter later is compared.

e. Click **Save**.

**Step 8.** To apply one or more of the dashboard's filters to one of its dashlets, do the following:

a. Select the dashlet and open its **Details** page to configure its underlying query.

**Note:** On the **Details** page, a field displays for each filter you have configured for the dashboard. Beneath each filter field is a text template for a "where" clause that refers to that filter. Because the syntax for a "where" clause must be precise, these templates are a useful starting point for incorporating the filter in to the dashlet's query.

b. Choose a filter and copy its template "where" clause text.

c. Paste the template text to the appropriate position for a "where" clause within the dashlet query string.

d. Edit the "where" clause so that it refers to a valid field within the query's data set.

**Note:** Take care to ensure that the resulting "where" clause precisely matches the spacing and syntax of the template provided. Symbols and spaces must all be placed correctly. If there is any error in the formatting of the clause, the query fails.

e. Repeat steps b, c, and d to add more filters to the query if required. Be sure to precisely follow the correct syntax for multiple filters in a query.

**Step 9.** Repeat step 8 for every dashlet on the dashboard that should be subject to the dashboard's filters.

**Step 10.** To apply dashboard filters to the dashlets on the dashboard designer page, do the following:

a. Enable the **Show filters bar** to turn on the dashboard designer page to display the filter fields on the dashboard designer page.

**Expected outcome**

The filters bar displays at the top of the dashboard's Layouts panel.

b. Click the filter icon on the filters bar to display a list of available filters, and click a filter in the displayed list.

c. Repeat step b to display additional filter fields if required.

d. Enter values into one or more of the displayed filter fields.

**Expected outcome**

All dashlets on the dashboard whose underlying queries use one of the filter values update to constrain their data to results matching the filter values.

## 5.7 Sharing dashboards

### About this task

When you create a new dashboard, it is displayed as a new Personal dashboard only in the Dashboards list for your user account. By default only you can see, use, modify, or delete it.

If you would like to share a dashboard for use by others, follow the steps in this procedure.

**Note:** Once you share a dashboard, it appears as a new, shared entry in the dashboard list. This shared copy can be modified by other users. Changes to the shared dashboard do not alter the original Personal dashboard.

**Note:** To view shared dashboards, users require URL Rule read permission to path '/core/user-storage/v2/shared/

**Note:** To publish and edit shared dashboards, users require URL Rule readWrite permission to path '/core/user-storage/v2/shared/

If you later modify a Personal dashboard that you previously published, you can share these changes by Publishing the dashboard again. This overwrites the previous Shared version of that dashboard.

### Procedure

**Step 1.** Use the **Main** navigation panel to select **Dashboards**.

**Step 2.** Select a Personal dashboard in the list.

**Step 3.** Use the **Table row actions** drop-down list to click **Publish**.

**Expected outcome**

A new copy of the selected dashboard is added to the Dashboards list, with the Type set to Shared.

# 6 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. Each EDA user can be granted resource access to specified namespaces, or cluster-wide. 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.

In the API responses, this base namespace is included in the `.metadata.namespace` field for non-namespaced resources and workflows.

In the UI, the `.metadata.namespace` value appears in the schema form split view (YAML/JSON) for non-namespaced resources and workflows.

## Namespaces in the EDA GUI

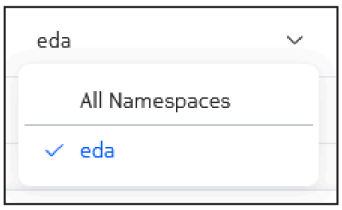
The top of every page in the EDA GUI includes a namespace selector. You must use this field to specify the namespace you are working in.

**Note:** The namespaces listed in the selector is limited to the namespaces that you have permission to access. Access to namespaces is granted by permissions configured by the EDA administrator. You have access to a namespace if you are a member of a group with a role of that namespace assigned.

You have access to every available namespace if you are:

- a member of a group with a ClusterRole resourceRule for any namespaced resource
- a member of a group with a ClusterRole urlRule for any namespaced API endpoint (that is, any API that takes namespace in the query parameter or path)
- a member of a group with any ClusterRole tableRule

*Figure 32: 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.
- If you have permission for non-namespace resources, these are displayed in the EDA GUI for any selected namespace.

**Note:** For EQL queries, **All Namespaces** must be selected to see results from the base namespace.

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.

When creating a resource, if a namespace is not selected, a message prompts you to select one for the autocomplete guidance to function.

## 6.1 Creating a namespace

### About this task

Only users with sufficient privileges can create a new namespace.

### Procedure

**Step 1.** From the **System Administration** navigation panel, click **Namespaces**.

**Step 2.** On the **Namespaces** page, click **Create**.

**Step 3.** Provide the following details for the namespace:

- Name
- Namespace (if none is selected)
- Labels
- Annotations

**Step 4.** Click **Commit** to commit your change immediately or click **Add To Transaction** to add this item to transactions to commit later.

# 7 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.

## 7.1 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 33: Resource metadata*


sc0226

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.

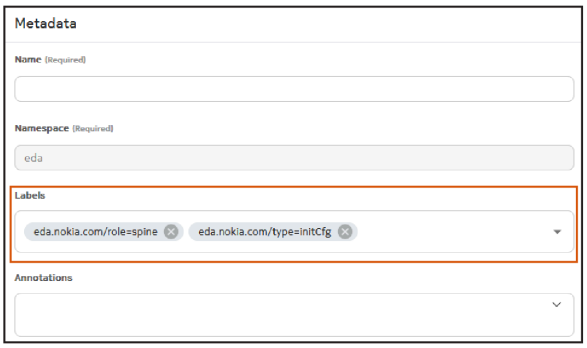
A label consists of two pieces of information: a key, and a value. Labels are limited to key-value pairs of small size and are designed for simple, static values. For example:

- 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 34: Selecting objects based on labels*




sc0227

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`.

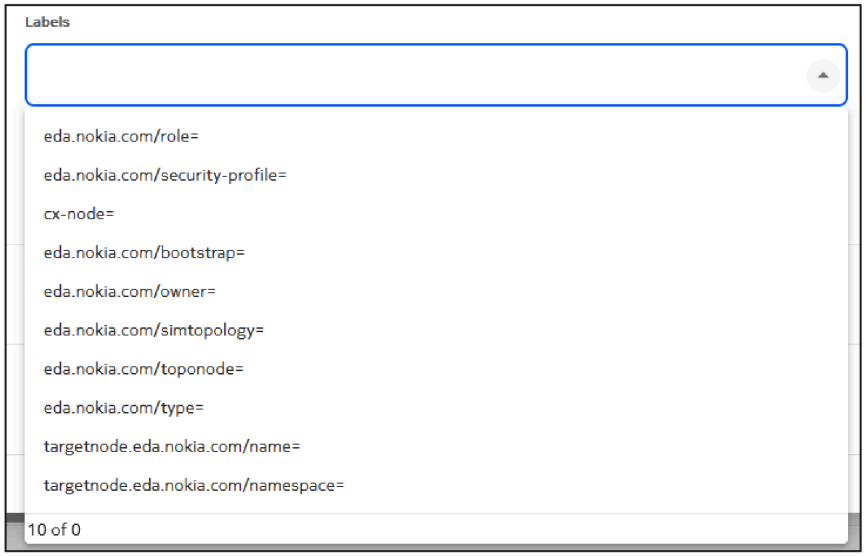
Some examples:

- `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

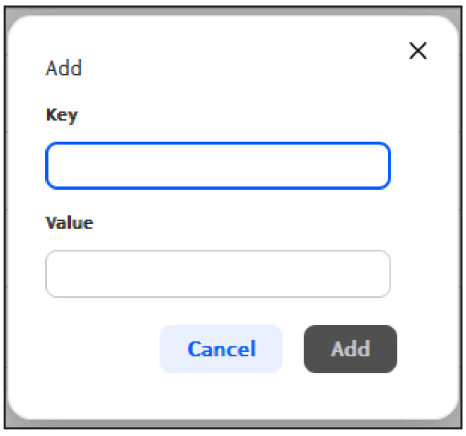
In the EDA GUI, where a **Label** field is present, you can enter a label by clicking in the **Label** field. This displays a list of available labels to choose from:




sc0228

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:



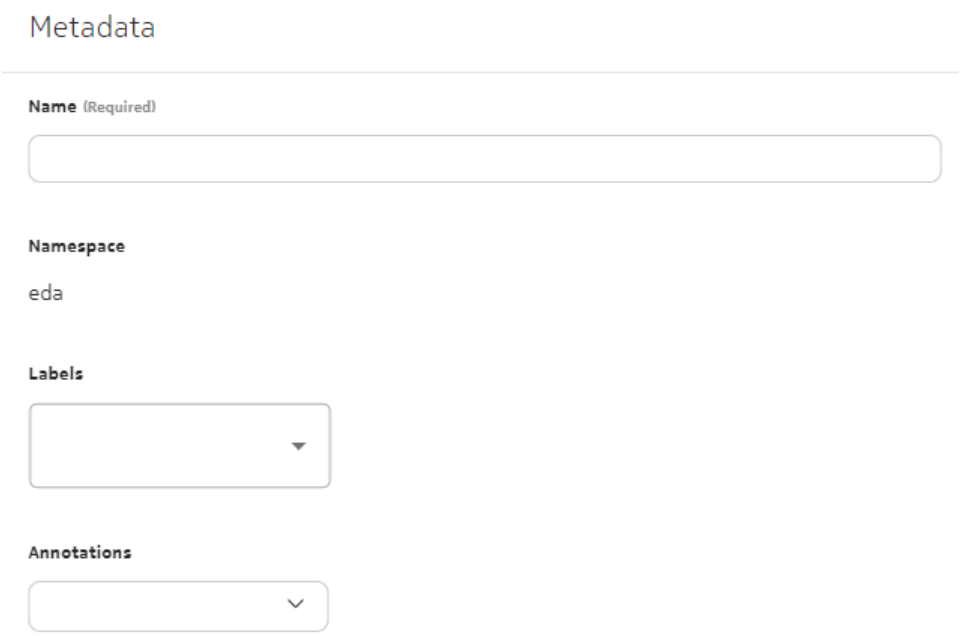

sc0230

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

## 7.2 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 35: Resource metadata*




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:

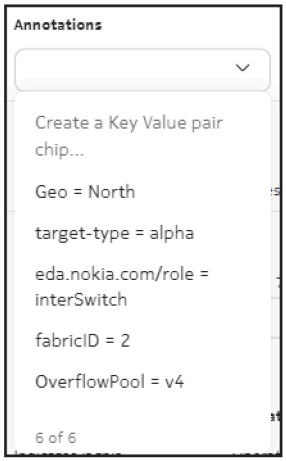
- kubectl.kubernetes.io/last-applied-configuration="JSON"
- author=team-name
- description="Stores the last applied configuration of a resource for use by kubectl apply"

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

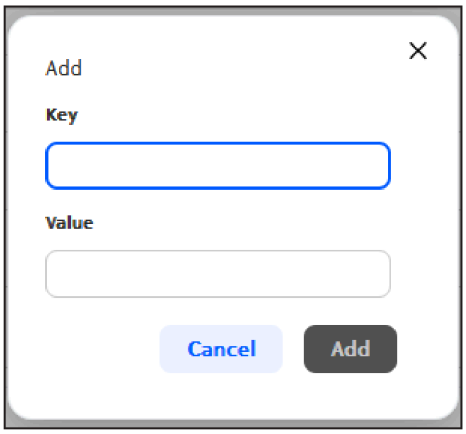
In the EDA GUI, where an **Annotation** field is present, you can enter an annotation by clicking in the **Annotation** field. This displays a list of available annotations to choose from:




*sc0263*

To use an existing annotation, select it from 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:




sc0230

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

## 7.3 Resource topologies

Because the volume of resources and their relationships within EDA is very large, it can be difficult to effectively grasp the relationship between one resources, and all of the other configured resources on which it somehow depends.

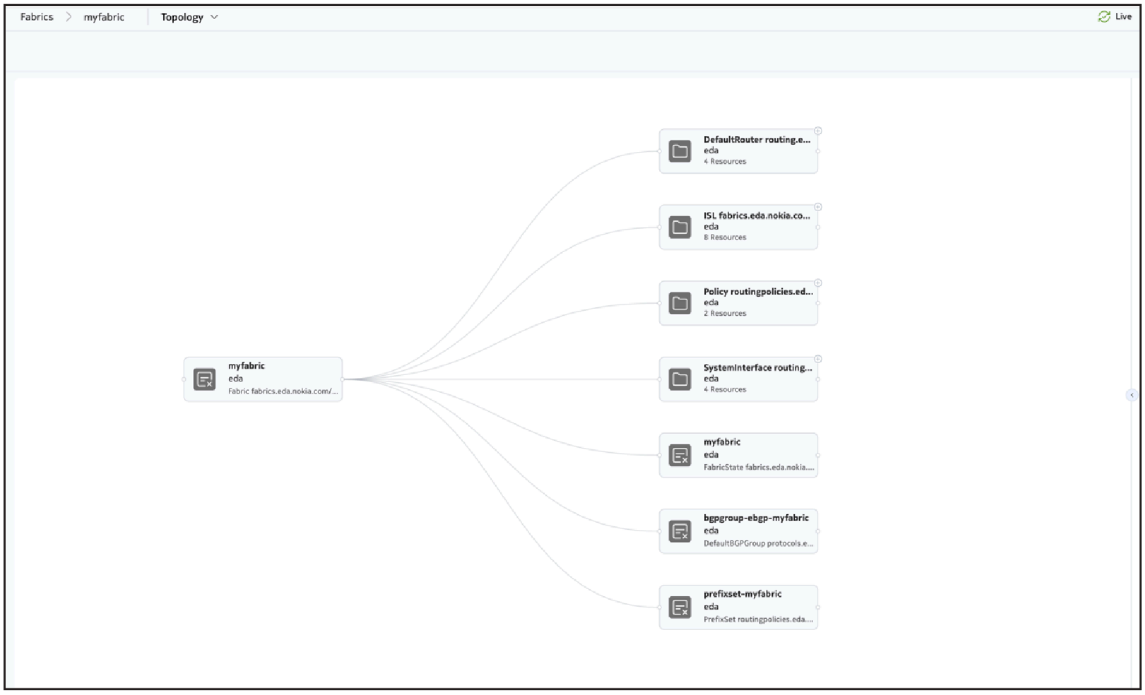
To help represent resources and their interconnections, EDA builds on its topology visualization framework by providing a Topology illustration. This illustration shows the selected resource, and the other EDA resources to which it is connected.

To see the topology view for a resource, open the **Details** view for an individual resource, and then select **Topology** from the drop-down list of available views.

For example, the following illustration shows the resource topology for a fabric. It shows not the fabric's physical topology, but its connection to the set of other resources configured within EDA:

- default routers
- ISLs
- routing policies
- system interfaces
- BGP groups
- prefix sets

*Figure 36: An example of a fabric resource topology*




*sc0310*

In the EDA UI, you can click any resource in the illustration to see more information in the Information panel.

# 8 Workflows

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 node, validate connectivity in a virtual network, 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. EDA uses CRDs to define the schema for the workflow input and output. EDA applications manifest tagged the `workflow` Boolean and include the relevant workflow to associate the CRD with the container image

EDA supports interactions with workflows through the following means:

- Through the EDA UI.
  - The **Workflows** button allows you to interact with workflows from anywhere in the UI.
  - **Workflow Definitions List** and **Workflows Executions** pages allow you to view and manage workflow definition lists and executed workflows.
  - From target resources, you can create and run workflows from the row actions menu.
- Through the API.
- Through the **edactl** command.
- Through Kubernetes: creating a resource whose CRD is marked as a workflow.

## FlowEngine

In EDA, workflows are supported through the FlowEngine - the controller behind the instantiation, status, and interaction with workflows.

On creating a new workflow, the FlowEngine:

- Validates the resources input against schema.
- Publishes a Kubernetes `Job` resource, which runs a container image provided in the corresponding `WorkflowDefinition`.
- Assigns a Flow ID to the workflow.
- Updates the status of the flow based on gRPC interactions.

Flow IDs are incremental. In the event of FlowEngine restart, previously executed or currently running flows are lost and a new Flow ID restarts at 1.

FlowEngine supports the following:

- Loading of new workflow definitions via EDA apps.
- Manual triggering of workflows.
- Reporting the status of executed workflows.

- General user interactions with workflows - the ability to block a flow and wait for user input.
- Workflow hierarchy - one workflow may be a parent of another.
- Automatic creation of workflow IDs.
- Artifacts - a workflow may return artifacts, such as tech support files, to the FlowEngine.
- Non-blocking behavior; up to 256 workflows can be executing at the same time.

**Note:** To avoid excessive memory use by FlowEngine, EDA enforces the following:

- Only 256 parent workflows are persisted.
- New workflows push out old workflows.
- There is a limit of 256 concurrently running workflows. New workflows are rejected if the system has reached this limit; actively running workflows are never dropped.
- This history includes stages and logs.
  This history persists for the lifetime of FlowEngine. It does not persist and does not remain after a restart.

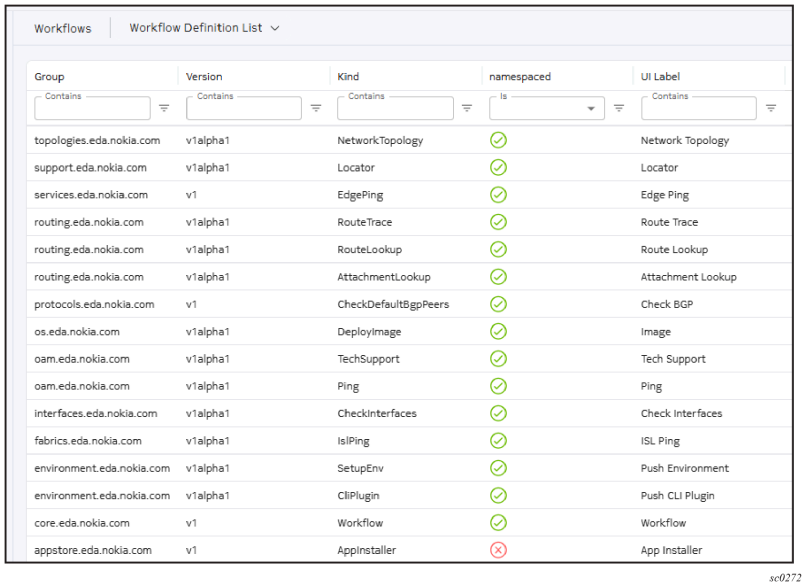
**Related topics**

*Workflows*

## 8.1 Workflow Definition List page

The **Workflow Definition List** page shows all available workflow definitions provided from EDA apps. From the **Main** navigation panel, click **Systems** under the **SYSTEM** group. Then, select **Workflow Definition List** from the drop-down list.

*Figure 37: **Workflow Definition List** page*




sc0272

The default workflow definitions shipped with the EDA apps are shown below.

*Table 20: Workflow definitions*

| Workflow definition | Purpose |
|---|---|
| App Installer | Used to install or delete apps. |
| Attachment Lookup | Used to look up attachments (where an address is attached in the network) on a set of nodes. The output shows the matching attachments, including the node, network instance, prefix, interface, and next hop group ID. |
| Check BGP | Checks the state and status of the BGP peers that match the selection criteria. |
| Check Interfaces | Used to check the state and status of the matched notes. Use interface selectors to select target interfaces on which to run this workflow. |
| Edge Ping | Used to initiate a ping to an edge interface resource; specify a gateway or edge mesh. |
| Image | Used to upgrade or downgrade software images on specified targets. This workflow can be used directly on a target or list of targets, or with selectors to select targets through labels. It also supports tranches, which are groups of targets that can be upgraded together. By default, the system runs a set of checks before and after the image change; you can upgrade this behavior by setting the **Checks** field.<br>This workflow also supports canary nodes, which are used to test images before a broader roll out. Canary nodes are upgraded before any other targets. To identify the canary nodes, use node selectors that match labels on TopoNode resources, including those in the list of nodes to be imaged. |
| ISL Ping | Used to ping inter-switch links (ISLs) to verify connectivity within a fabric. You can specify a list of fabrics, ISLs, or selectors for both to match ISLs. This workflow shows the results of the pings, including the status of each ISL. |
| Locator | Typically used to guide on-site technicians to the correct target requiring maintenance; enables the LED locator for a target. |
| Network Topology | Allows you to perform create, replace, and delete operations on the specified topology.<br>• Create - creates/updates the topology resources based on the provided specifications.<br>• Replace - replaces the resources matched by name with the provided specifications.<br>• ReplaceAll - first removes all existing topology resources and then creates new ones based on the provided specifications.<br>• Delete resources matched by name. |

| Workflow definition | Purpose |
|---|---|
| | • Delete all - delete all topology resources found in the specified namespace. |
| Ping | Used to initiate a ping to an address on a node or a set of nodes |
| Push CLI Plugin | Used to push a CLI plug-in to a node.<br>For SR Linux, the plug-in that you specify must include the `.py` extension, without leading slashes, for example, `"myplugin.py"`. |
| Push Environment | Used to set up the global environment on a node. For SR Linux, this results in an overwrite of the `/etc/opt/srlinux/env` file. |
| Route Lookup | Used to look up routes on a set of nodes. The output shows the matching route and the set of ingress interfaces used to reach it. |
| Route Trace | Used to trace routes for specified targets. |
| Tech Support | Used to generate technical support packages for a node or set of nodes; typically used for debugging. |
| Workflow | A generic workflow definition. This workflow is used by some workflows to create subflows without a predefined schema. |

**Related topics**

*Common data grid actions menu*

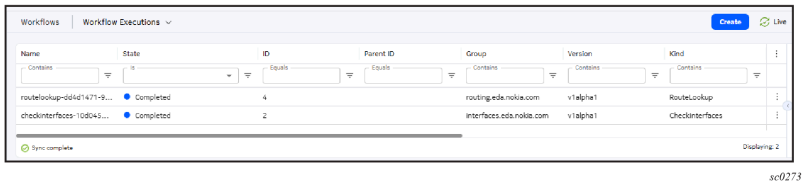
## 8.2 Workflow Executions page

The **Workflow Executions** page shows all the workflows that have been executed.

From the **Main** navigation panel, click **Systems** under the **SYSTEM** group. Then, select **Workflow Executions** from the drop-down list.

Any workflow waiting for user input is highlighted in yellow.

*Figure 38: The **Workflow Executions** page*




sc0273

From **Workflow Executions** view, you can:

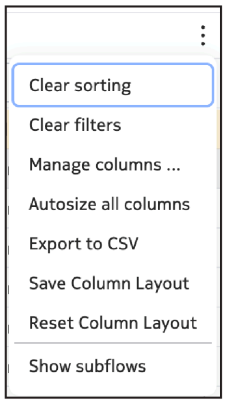
- provide additional input for a workflow, if user input is required
- create and run a new workflow
- cancel a running workflow
- display the **Summary** page for a workflow

Click the **Table row actions** menu for a workflow and select **Details** to display the **Summary** page for the workflow.

## Displaying subflows

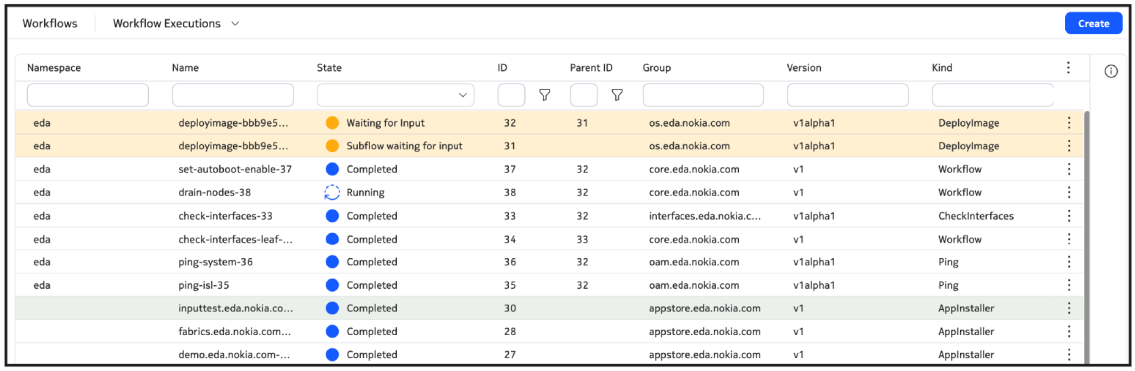
By default, only top-level workflows are displayed. To display subflows, click the **Table settings & actions** icon and select **Show Subflows**.

*Figure 39: Subflows*




sc0274

*Figure 40: Workflow executions with subflows*




sc0275

### Related topics

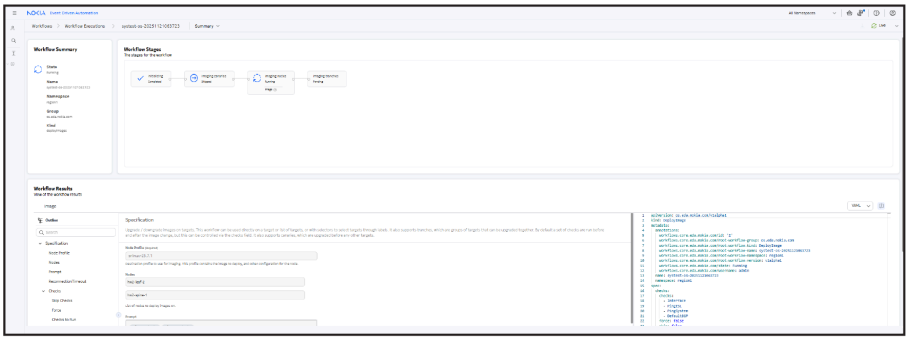
*Workflows*
*Common data grid actions menu*
*Running a workflow from the Workflow Executions page*

## 8.3 Workflow Summary page

The **Summary** page provides details about a workflow execution. Click any workflow in the **Workflow Executions** page to display its summary.

*Figure 41: Workflow Summary page*

The following example is for a DeployImage workflow.



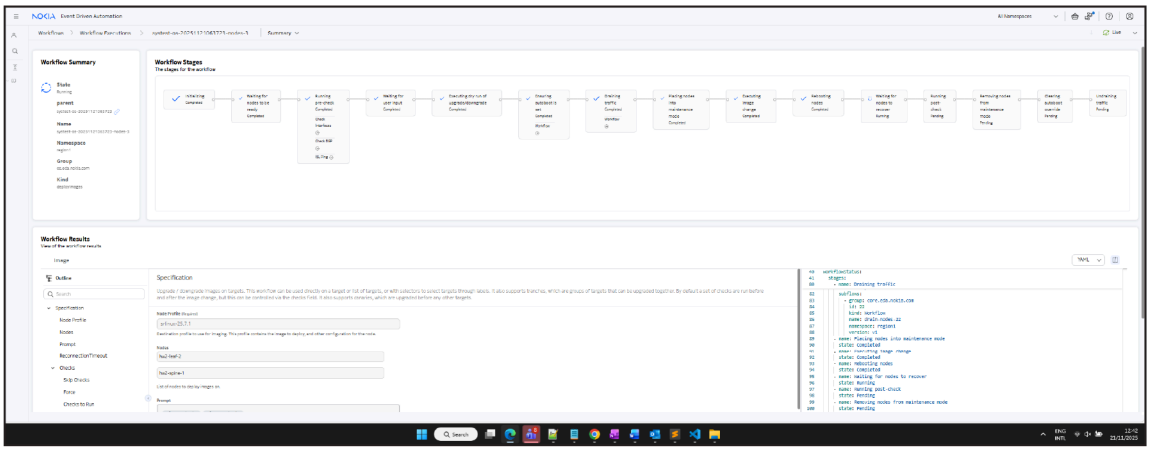
sc0276

The **Workflow Summary** panel provides the status of the workflow, name, namespace, category, and type of workflow.

- Status of the workflow, which can be one of the following:
  - Waiting to start
  - Running
  - Completed
  - Failed
  - Terminated
  - Waiting for input
- Stages of the workflow.
  The **Workflow Stages** panel shows the progress of workflow and the status of the workflow as it passes through the stages.
- The **Workflows Results** panel provides details of the workflow results, including the specification (input) and status (output) of the workflow that was executed.

You can display more information about a stage in a workflow when an arrow is present in its box. In the preceding example, if you click the arrow in the **Imaging nodes** box, the following screen displays:

*Figure 42: Details for Imaging node stage*

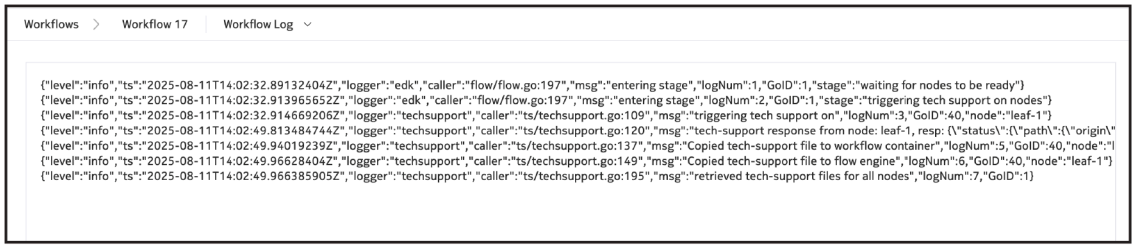


*sc0306*

## Workflow logs

The **Workflow Logs** view display logs from the workflow container. From the **Summary** drop-down list, select **Workflow Log**. The logs are used for troubleshooting and debugging purposes.

*Figure 43: Sample workflow log*




*sc0277*

## Workflow artifacts for download

The execution of some workflows produce artifacts that you can download. For example, the Tech Support workflow creates a `tech-support .zip` file. If a file is available for download, the download button is visible from the **Workflow Summary** page. Click it to download the artifact and save the file locally.

**Note:** This capability is currently supported only on SR Linux deployments.

You can also download artifacts using the **edactl** command; see Managing workflows with edactl.

**Related topics**

*Workflows*

## 8.4 Workflow creation

You can create a workflow using one of the following procedures:

- Running a workflow from the Workflow Definition List page
- Running a workflow from the Workflow Executions page
- Triggering a workflow from the resource action menu

### 8.4.1 Running a workflow from the Workflow Definition List page

#### About this task

You can run a workflow by creating another instance of that kind of workflow and running it.

#### Procedure

**Step 1.** Select the workflow definition that you want to run by double-clicking it or clicking **Run** from the **Table row actions** icon.

**Step 2.** In the form that displays, fill in the values for the workflow.

The contents of the form vary depending on the workflow definition selected. The EDA UI auto-generates a unique name for the workflow execution. You can override this name with a custom name.

**Step 3.** Click **Run**.

### 8.4.2 Running a workflow from the Workflow Executions page

#### About this task

You can create a new workflow by selecting the type of workflow that you want or by duplicating an existing workflow and updating the prepopulated specifications.

#### Procedure

**Step 1.** From the **Main** navigation panel, click **Workflows**.

**Step 2.** Click **Workflows Executions** from the **Workflows** drop-down list.

**Step 3.** You can create a new workflow or duplicate an existing one.

- Create a new workflow.
  - **a.** Click **Create**.
  - **b.** Select the workflow that you want to run from the drop-down list.
  - **c.** In the form that opens, enter the specifications for the workflow.
    The contents of the form vary depending on the workflow definition that selected. EDA auto-generates a unique name for the workflow execution. You can override this name with a name of your choice.
- Duplicate an existing workflow.

a. Locate the workflow that you want to duplicate and click **Duplicate** from its **Table row actions** menu.
The prepopulated form displays.

b. Update the specifications for the workflow as needed.

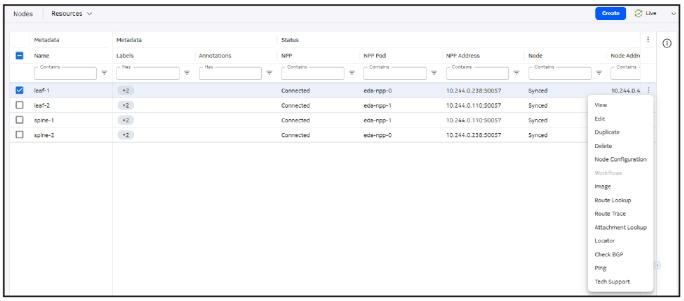
**Step 4.** When you are finished entering specifications for the workflow, click **Run**.

### 8.4.3 Triggering a workflow from the resource action menu

#### About this task

You can run a workflow from the **Row action menu** of target resources.
Workflow definitions define some types of resources as subjects. These workflows are listed in the action menu of the relevant resources. For example, the Ping workflow accepts a node as subject and can be triggered from the node action menu:

*Figure 44: Workflow options*




sc0278

Some workflows allow you to select multiple resources in the same workflow. For example, for the Image workflow, you can identify multiple Node resources. You cannot include resources from multiple namespaces in the same workflow input. The bulk workflow actions function is disabled when the UI page is in set to **All Namespaces**.

The following example executes the Ping workflow.

#### Procedure

**Step 1.** From the **Main** navigation panel, under **TARGETS**, click **Nodes**.

**Step 2.** Select **Resources** from the drop-down list.

**Step 3.** Locate the resource and select **Ping** from the **Table row actions** menu.

**Step 4.** In the form that opens, fill the specifications for the workflow, such as the destination address.
The contents of the form vary depending on the workflow definition that you select.

**Step 5.** Click **Run**.

## 8.5 Managing workflows with edactl

You can use the **edactl** command to provide input so a workflow can proceed or to query EDA about workflows.

### Providing input to workflows

Some workflows may require user input to allow the workflow to proceed. You can use the following commands to handle workflows that require user input:

- To find workflows awaiting input, use the following command and look for status 'WAITING_FOR_INPUT':

```
edactl workflow get -A -a
```

- To acknowledge a workflow and allow it to continue, use the following command:

```
edactl workflow ack <id>
```

  For example, to acknowledge the workflow whose ID is 10:

```
edactl workflow ack 10
```

- To terminate a workflow, use the following command:

```
edactl workflow nack <id>
```

  For example, to terminate the workflow whose ID is 20:

```
edactl workflow nack 20
```

### Querying EDA

Use the following **edactl** commands to query EDA about workflows:

- To view all workflows, use the following command:

```
edactl workflow get -A
```

  For example:

```
edactl workflow get -A
ID NAMESPACE NAME TYPE STATUS
1 eda-system bulkapps-eda.nokia.com app-installer COMPLETED
2 eda-system bulkapps-eda.nokia.com app-installer FAILED
```

- To view details of a specific workflow, use the following command:

```
edactl workflow get <id>
```

  For example:

```
edactl workflow get 1
ID: 1
Namespace: eda-system
Name: bulkapps-eda.nokia.com
Status: COMPLETED
Workflow Steps:
↓ init
↓ Fetching
↓ Verifying
↓ Committing
```

```
↓ Applying
↓ Installed
```

- To view logs for a workflow, use the following command:

  ```
  edactl workflow logs <id>
  ```

  For example: edactl workflow logs 20

- To tail log output of a running workflow, using following command:

  ```
  edactl workflow logs <id> --follow
  ```

- To list files associated with a specific workflow, use the following command:

  ```
  edactl workflow artifacts <id>
  ```

  For example:

  ```
  edactl workflow artifacts 2
  Artifacts available for the workflow:
        tech-support-20250207_050610-mv1nd01-spine-1.zip
  root in on eda-toolbox-6f6c686487-xdks4 /eda
  ```

- To download all files associated with the workflow in present working directory, use the following command:

  ```
  edactl workflow artifacts <id> download
  ```

  For example:

  ```
  edactl workflow artifacts 2 download
  Downloading artifacts to: /eda
  ```

- To download all the files associated with the workflow in the `/tmp/` directory, use the following command:

  ```
  edactl workflow artifacts <id> download --to /tmp/
  ```

  For example:

  ```
  edactl workflow artifacts 2 download --to /tmp/
  Downloading artifacts to: /tmp
  tech-support-20250207_050610-mv1nd01-spine-1.zip 100% [===============] (5.9/5.9 MB, 98 MB/
  s)
  ```

- To download a single file associated with the workflow in the `/tmp/` directory, use the following commands:

  ```
  edactl workflow artifacts <id> download --to /tmp/ --from <file name>
  ```

  or

  ```
  edactl workflow artifacts 2 download –from <file name>
  ```

For example:

```
edactl workflow artifacts 2 download --from tech-support-20250207_050610-mv1nd01-spine-1.zip
 --to /tmp/ Downloading artifacts to: /tmp tech-support-20250207_050610-mv1nd01-spine-1.zip
 100% [===============] (5.9/5.9 MB, 104 MB/s) root in on eda-toolbox-6f6c686487-xdks4 /eda
```

# 9 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 `resource`, `group`, and `kind` fields.

Alarms 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

The EDA GUI includes several summary views of alarms known to EDA:

- An alarm summary is displayed on the EDA home page.
- A more detailed summary of alarms affecting key EDA components (clusters, Git servers, App catalogs and registries) is displayed on the main **Alarms Summary** page
- The Alarms list displays a list of all active alarms

From the Alarm list, 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.

**Note:** You can still view suppressed alarms by choosing "Show all alarms" from the Alarm List Table Settings and Actions menu.

- 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.

### Alarm definition resources

EDA core services and EDA applications document their supported alarms as alarm definition resources. These resources are available via the query `.namespace.resources.cr.core_eda_nokia_com.v1.alarmdefinition`. This query allows operators to retrieve a list of supported alarms that is specific to their deployed apps and versions.

The alarm definition specifications include the following fields:

*Table 21: Alarm definition specifications*

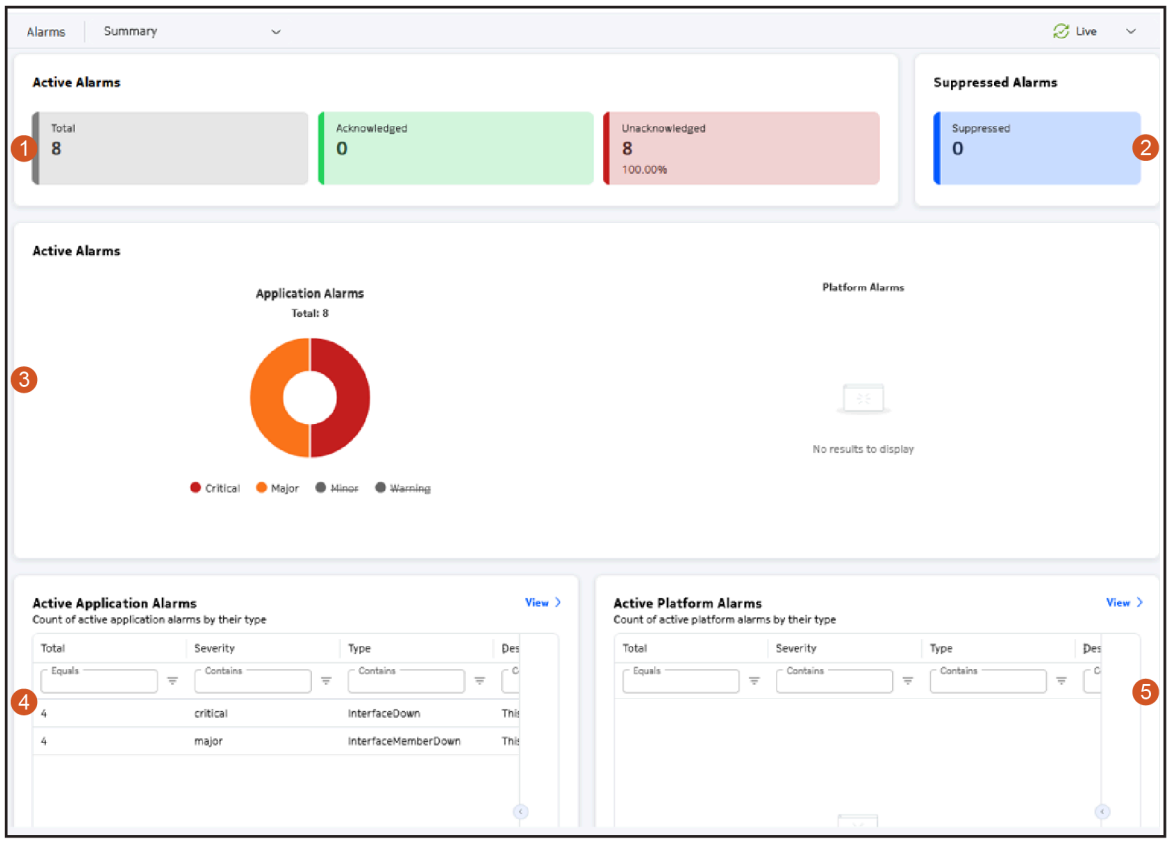
| Specification | Description |
|---|---|
| SourceGroup | Specifies the group of the resource that raises the alarm, for example, `interfaces.eda.nokia.com`.<br>Identifies the application that raises the alarm. EDA core alarms have the source group `core.eda.nokia.com`. |
| Type | Specifies the alarm type, for example, `Interface Down`. |
| Description | Specifies the description of the alarm. |
| ClusterSpecific | If this specification is true, the alarm is raised against a specific EDA cluster.<br>For example, a raised DeploymentDegraded alarm specifies which EDA cluster (active or standby) has a Kubernetes Deployment with one or more replicas that are not running. |
| Severity (optional) | Specifies alarm severity.<br>If not set, the alarm severity is variable. For example, the CPUThresholdExceeded alarm increases severity based on the CPU utilization value. |
| Kind and Group (optional) | Specifies the Kind and Group of the resource which the alarm is associated with.<br>For example, an InterfaceDown alarm is raised against an Interface resource. |

| Specification | Description |
|---|---|
| | The Kind is `Interface` and the Group is `interfaces.eda.nokia.com`. |

**Note:** An alarm definition is unique to the combination of SourceGroup and Type specifications. For example, both the Routing and Services apps can raise an alarm type BFDSessionDown. These are separate alarm definitions.

## 9.1 The Alarms Summary

*Figure 45: The Alarms Summary page*




sc0232

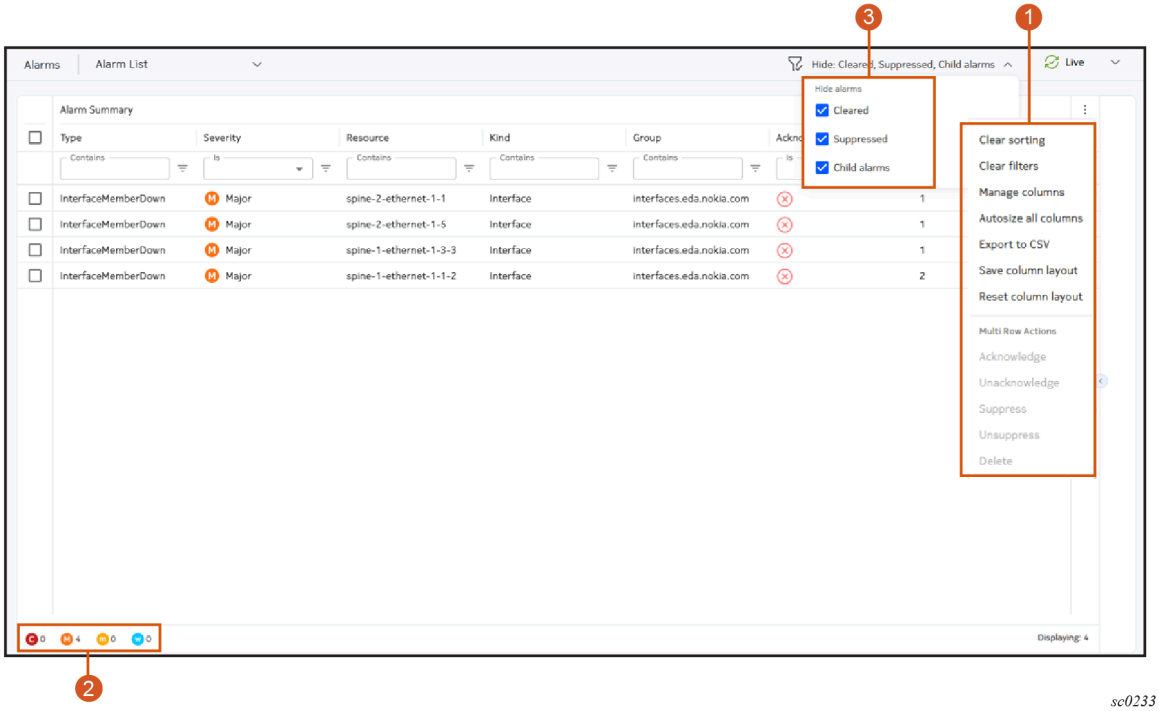
*Table 22: Elements of the **Alarms Summary** page*

| # | Dashlet | Description |
|---|---|---|
| 1 | Active Alarms count | The Active Alarms count dashlet displays a total count of alarms affecting EDA applications and the EDA platform itself. This dashlet also displays |

| # | Dashlet | Description |
|---|---|---|
| | | the count of acknowledged and unacknowledged alarms. |
| 2 | Suppressed Alarms | The Suppressed Alarms dashlet displays the count of suppressed alarms. |
| 3 | Active Alarms charts | The Active Alarms charts dashlet breaks the Application Alarm and Platform Alarm counts down by severity and displays them in charts. |
| 4 | Active Application Alarms | Building on the data displayed in the Active Alarms panel, the Active Application Alarms dashlet lists the active alarms affecting EDA applications, their severity, and their type.<br>Clicking the **View** link opens the Alarms List. |
| 5 | Active Platform Alarms | The Active Platform Alarms dashlet lists the active alarms affecting the EDA application, their severity, and their type.<br>Clicking the **View** link opens the Alarms List. |

## 9.2 The Alarms list

*Figure 46: The Alarms list*




sc0233

*Table 23: Elements of the Alarms list*

| # | Name | Function |
|---|---|---|
| 1 | Alarms menu | The Alarms menu includes:<br>• common table controls<br>• multi-row actions unique to alarms<br>• special actions unique to the alarms list |
| 2 | Alarm count | Displays the number of current alarms of various severities. |
| 3 | Alarm quick filters | Allows you to hide alarms using the following filters:<br>• **Cleared** – filters out cleared alarms<br>• **Suppressed** – filters out suppressed alarms<br>• **Child alarms** – filters out child alarms<br>These filters are all selected by default. |

### Columns

The list of alarms displays the following columns by default.

*Table 24: Default alarm display columns*

| Column | Description |
|---|---|
| Namespace | Indicates the namespace to which the alarm belongs.<br>Alarms that are not specific to a namespace, such as platform certificate alarms, do not have or display a namespace value. |
| Type | The alarm type, as defined by the alarm itself. For example, InterfaceDown. |
| Severity | The importance of the alarm, as defined by the alarm itself. Supported severities are:<br>• Critical<br>• Major<br>• Minor<br>• Warning<br>You can click this column to sort by severity level. |
| Resource | Indicates the name of the resource that this alarm is present on.<br>• 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.<br>• A resource may also be an instance of a resource in the Kubernetes cluster where EDA is deployed. For example, the resource may be the name of a Pod. |
| Kind | Indicates the kind of resource the alarm is present on. |

| Column | Description |
|---|---|
| | • For example, this is set to Fabric if an alarm is raised against a Fabric resource.<br>• The kind may also be a component with the EDA core - for example, Config Engine or StateEngine. |
| Group | Indicates the group of the resource the alarm is present on.<br>For example, this is set to fabrics.eda.nokia.com if an alarm is raised against a Fabric. |
| Acknowledged | Indicates whether the alarm has been acknowledged (True or False) |
| Occurrences | The number of occurrences for the alarm. |
| 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. |

The following alarms can be optionally displayed in the list of alarms by selecting **Manage columns** from the Alarms menu.

*Table 25: Non-default alarm display columns*

| Column | Description |
|---|---|
| Name | Indicates the name of the alarm. |
| Description | The description of the alarm from the alarm's encoded Description field. |
| Cleared | Whether the alarm has been cleared by an operator. Possible values are:<br>• True<br>• False<br>This column is visible if the **Active only** filter is turned off. |
| Cluster Member | For EDA platform alarms, the EDA cluster member to which the alarm applies. |
| Probable Cause | The probable cause of the alarm, from the alarm's encoded Probable Cause field. |
| Remedial Action | The suggested remedial action to resolve the alarm, from the alarm's encoded Remedial Action field. |
| Acknowledged Until | If the alarm has been temporarily acknowledged, this indicates the date and time at which the acknowledgement expires. |
| Last Acknowledged | Indicates the date and time when the most recent acknowledgement occurred for this alarm. |
| Last Acknowledged By | Indicates the user who last acknowledged the alarm. |
| Suppressed Until | If the alarm has been temporarily suppressed, this indicates the date and time at which the suppression expires. This column is visible if the **Hide suppressed** filter is turned off. |
| Suppressed | Indicates whether the alarm has been suppressed. This column is visible if the **Hide suppressed** filter is turned off. |

| Column | Description |
|---|---|
| Last Suppressed | Indicates the date and time when the most recent suppression occurred for this alarm. |
| Last Suppressed By | Indicates the user who last suppressed the alarm. |
| Source Resource | The EDA-managed resource from which the alarm originates. |
| Source Kind | The kind of resource from which the alarm originates. |
| Source Group | The group of the resource from which the alarm originates. |
| JS Paths | The EDB entry which triggered the raising or clearing of the alarm, in JSPath notation. For example, if the alarm pertains to an interface operational state, this may be the JSPath: .node{.name=="spine-1"}.srl.interface{.name=="ethernet-1/14"} |
| Parent Alarms | Indicates whether the alarm is associated with one or more parent alarms. It is common for alarms to have one or more parents. For example, an Interface Degraded alarm may be caused by one or more of its component members being down; that condition is itself the subject of a separate alarm. This column is visible if the **Root cause only** filter is turned off. |
| Targets Affected | Indicates the number of child items which show the correlation between an alarm and the target it affects. |

**Related topics**

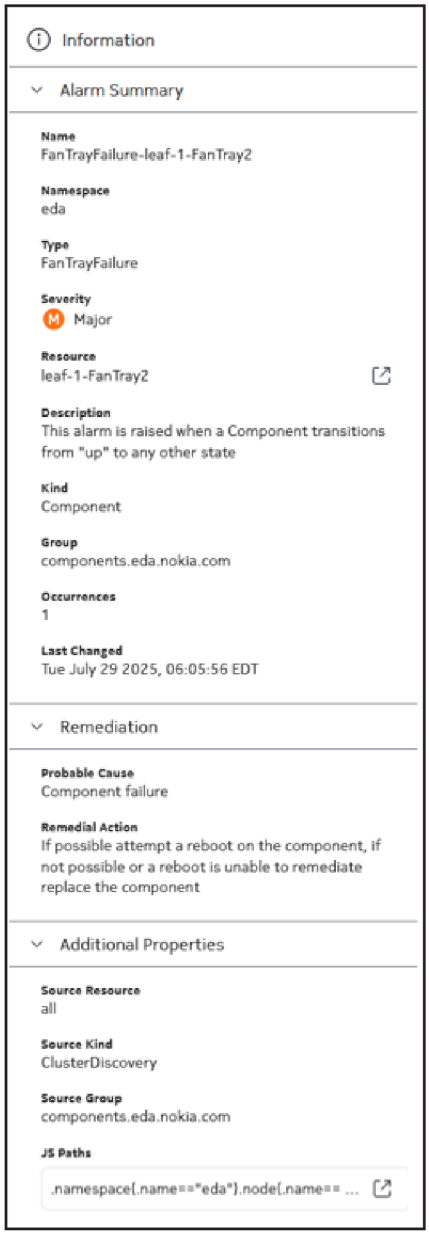
*Working with data grids*

*Namespaces*

## 9.3 The Alarms Information panel

The **Alarms list** page contains an **Information** panel that you can open by clicking the Expand/Contract control at the middle right of the page. This **Information** panel appears throughout the EDA GUI, for details see Information panel. The following figure shows an example of an expanded Alarms Information panel for a selected alarm.

*Figure 47: The expanded Alarms Information panel*




*sc0264*

The **Information** panel displays information about any selected alarm in the Alarms list. This information includes the following grouped and collapsible fields:

- Alarm Summary – includes Name, Namespace, Type, Severity, Cleared, Resource, Description, Kind, Group, Occurrences, Acknowledged, and Last Changed fields
- Remediation – includes Parent Alarms, Probable Cause, and Remedial Action fields
- Additional Properties – includes Acknowledged Until, Last Acknowledged, Suppressed, Suppressed Until, Last Suppressed, Source Resource, Source Kind, Source Group, and JS Path fields

Fields that are empty or not applicable are hidden.

The following table describes the navigation from the **Information** panel to Parent Alarms and Resource locations.

*Table 26: Information panel navigation details*

| Location | Description |
|---|---|
| **Alarm Summary → Resource** | Displays the resource information for the selected alarm.<br>Click **Go to resource** to view the details of the alarm resource. |
| **Remediation → Parent Alarms** | Displays the parent alarm of the selected alarm.<br>Click **Go to parent alarm** to display the **Details** page for the parent alarm of the selected alarm. |
| **Additional Properties → JS Paths** | Displays the JS path for the selected alarm.<br>Click **Go to Query Builder** to display the **Query Builder** page for the selected alarm. |

## 9.4 Sample core alarms

*Table 27: 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 are correct, and any offered certificates are trusted. |
| • server-name is the name of the Git server hosting the repository, for example primary.<br>• repo-type is the type of repository, one of Apps, Backup, Identity, Security, Catalog, UserSettings.<br>• source is the name of the pod that raised the alarm, for example eda-se-1.<br>• source-kind is one of ConfigEngine, StateEngine, AppStore.<br>• cluster is the name of the cluster member that this alarm was raised in, populated via setting the clusterSpecific flag. | |

| Property | Description |
|---|---|
| • server-uri/remote-path is the combination of the server URI and remote path as defined in the Engine Config. | |

*Table 28: 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 are correct (typically using no proxy for inter-cluster HTTPS), and certificate validity. |
| • service is the common name of the destination pod, for example eda-npp-leaf-1-1.<br>• source is the name of the source pod that raised the alarm, for example eda-sc-1.<br>• source-kind is one of APIServer, ConfigEngine, StateEngine, AppStore.<br>• cluster is the name of the cluster member that this alarm was raised in, populated via setting the clusterSpecific flag. | |

*Table 29: Pod Not Running*

| Property | Description |
|---|---|
| Name | PodNotRunning-<cluster>-<pod> |
| Severity | Critical |
| Description | Pod "<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. |
| Probable cause | Kubernetes controller or registry reachability issues, worker node failure, initial instantiation. |
| Remedial action | 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". |
| • cluster is the name of the cluster member that this alarm was raised in, populated via setting the clusterSpecific flag.<br>• pod is set to the name of the pod that is not running, but should be.<br>• config-engine is set to the name of the ConfigEngine pod that raised the alarm. | |

*Table 30: Deployment Degraded*

| Property | Description |
|---|---|
| Name | DeploymentDegraded-<cluster>-<deployment> |
| Severity | Critical |
| Description | 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. |
| Probable cause | Kubernetes infrastructure issues, worker node failure, initial instantiation. |
| Remedial action | 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". |
| • cluster is the name of the cluster member that this alarm was raised in, populated via setting the clusterSpecific flag.<br>• pod is set to the name of the pod that is not running, but should be.<br>• config-engine is set to the name of the ConfigEngine pod that raised the alarm. | |

*Table 31: 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.<br>• pod is set to the name of the pod that is not running, but should be.<br>• config-engine is set to the name of the ConfigEngine pod that raised the alarm. | |

*Table 32: NPP Down*

| Property | Description |
|---|---|
| Name | NPPDown-<cluster>-<npp> |

| Property | Description |
|---|---|
| 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.<br>• npp is set to the name of the destination NPP.<br>• config-engine is set to the Pod name of the ConfigEngine that raised the alarm. | |

*Table 33: 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.<br>• pool-name is the name of the allocation pool, for example ipv4-pool.<br>• pool-instance is the name of the instance of the pool, for example global.<br>• severity is the severity of the alarm, which increases based on which threshold has been breached.<br>• threshold is the value of the threshold that has been breached, for example 80.<br>• config-engine is the pod name of the ConfigEngine raising the alarm. | |

*Table 34: State Engine Reachability Down*

| Property | Description |
|---|---|
| Name | StateEngineReachabilityDown-<state-engine>-<state-controller> |

| Property | Description |
|---|---|
| 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 is not receiving 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 are 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.<br>• state-controller is the name of the State Controller pod that raised the alarm, for example eda-sc-1. | |

## 9.5 Viewing alarms

### About this task

Follow this procedure to view the set of alarms known to EDA.

**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.** In the EDA main menu, select **Alarms** to open the **Alarms** page.

**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.

## 9.6 Acknowledging an alarm

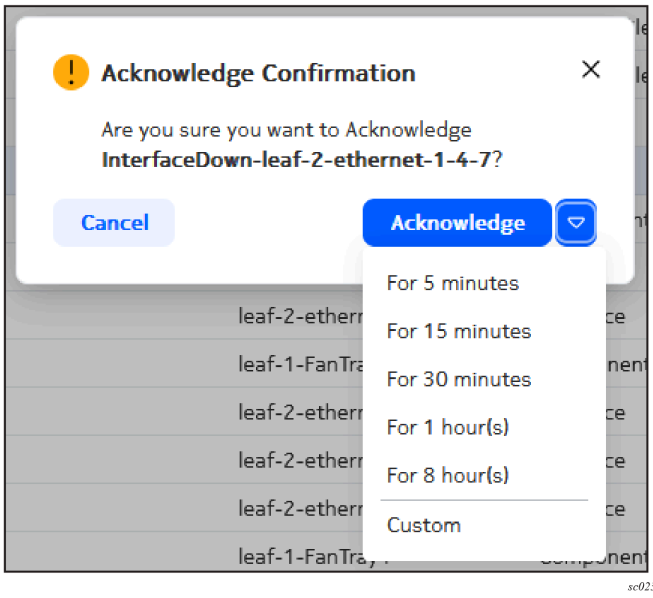
### About this task

When you acknowledge an alarm, you can:

- Acknowledge the alarm permanently.
- Acknowledge the alarm temporarily, after which the alarm returns to its unacknowledged state.

When you acknowledge an alarm temporarily, you can choose from a list of predefined periods, or select a specific time and date for the acknowledgement to expire.

*Figure 48: Temporarily acknowledging an alarm*




### Procedure

**Step 1.** In the EDA main menu, select **Alarms** to open the **Alarms** page.

**Step 2.** Find the alarm in the list using the sorting and filtering controls.

**Step 3.** At the right side of the row, click the **Table row actions** button.

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

**Step 5.** Optionally, you can choose to acknowledge the alarm only temporarily by doing either of the following:

- Click the drop-down control and click one of the standard periods displayed.
- Click the drop-down control, then click **Custom**, and in the resulting window select a date and time for the acknowledgement to expire.

**Step 6.** Click **Acknowledge** to complete the acknowledgement of the alarm.

## 9.7 Acknowledge multiple alarms

### About this task

When you acknowledge multiple alarms, you can:

- Acknowledge the alarms permanently
- Acknowledge the alarms temporarily, after which all of the selected alarms return to their unacknowledged state.

When you acknowledge alarms temporarily, you can choose from a list of predefined periods, or select a specific time and date for the acknowledgement to expire.

### Procedure

**Step 1.** In the EDA main menu, select **Alarms** to open the **Alarms** page.

**Step 2.** Use the sorting and filtering controls to display the necessary set of alarms in the list.

**Step 3.** 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 4.** At the upper right of the **Alarms** page, click the **Table settings & actions** button.

**Step 5.** Select **Acknowledge** from the list.

**Step 6.** Optionally, you can choose to acknowledge the alarm only temporarily by doing either of the following:

- Click the drop-down control and click one of the standard periods displayed.
- Click the drop-down control, then click **Custom**, and in the resulting window select a date and time for the acknowledgement to expire.

**Step 7.** Click **Acknowledge** to complete the acknowledgement of the selected alarms.

## 9.8 Deleting a single alarm

### Prerequisites

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

### Procedure

**Step 1.** In the EDA main menu, select **Alarms** to open the **Alarms** page.

**Step 2.** Find the alarm in the list using the sorting and filtering controls.

**Step 3.** At the right side of the row, click the **Table row actions** button.

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

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

**Step 5.** Click **Confirm** to complete the acknowledgement.

## 9.9 Deleting multiple alarms

### Prerequisites

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

### Procedure

**Step 1.** In the EDA main menu, select **Alarms** to open the **Alarms** page.

**Step 2.** Use the sorting and filtering controls to display the necessary set of alarms in the list.

**Step 3.** 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 4.** At the upper right of the **Alarms** page, click the **Table settings & actions** button.

**Step 5.** Click **Delete** in the list.

**Step 6.** 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.

## 9.10 Suppressing a single alarm

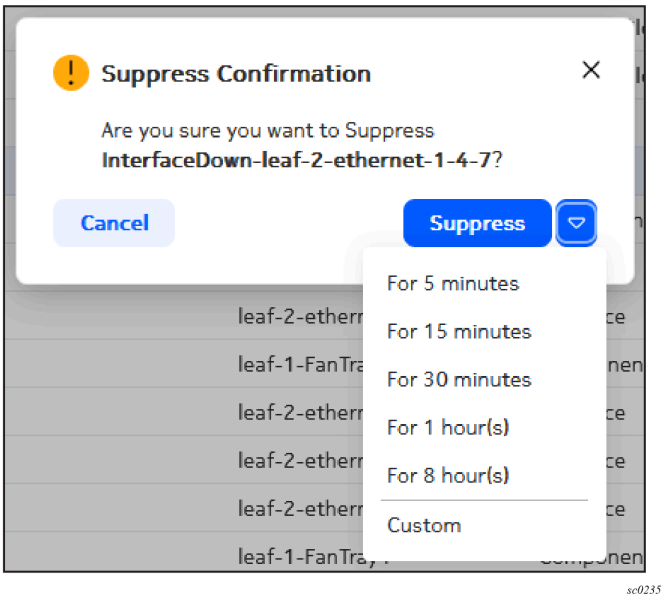
### About this task

When you suppress an alarm, you can:

- Suppress the alarm permanently.
- Suppress the alarm temporarily, after which the alarm returns to its unsuppressed state.

When you suppress an alarm temporarily, you can choose from a list of predefined periods, or select a specific time and date for the suppression to expire.

*Figure 49: Temporarily acknowledging an alarm*




### Procedure

**Step 1.** In the EDA main menu, select **Alarms** to open the **Alarms** page.

**Step 2.** Find the alarm in the list using the sorting and filtering controls.

**Step 3.** At the right side of the row, click the **Table row actions** button.

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

**Step 5.** Optionally, you can choose to suppress the alarm only temporarily by doing either of the following:

- Click the drop-down control and click one of the standard periods displayed.
- Click the drop-down control, then click **Custom**, and in the resulting window select a date and time for the suppression to expire.

**Step 6.** Click **Confirm** to complete the alarm suppression.

**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.

## 9.11 Suppressing multiple alarms

### About this task

Just as with single alarms, when you suppress multiple alarms, you can:

- Suppress the alarms permanently

- Suppress the alarms temporarily, after which all of the selected alarms return to their unsuppressed state.

When you suppress alarms temporarily, you can choose from a list of predefined periods, or select a specific time and date for the acknowledgement to expire.

### Procedure

**Step 1.** In the EDA main menu, select **Alarms** to open the **Alarms** page.

**Step 2.** Use the sorting and filtering controls to display the necessary set of alarms in the list.

**Step 3.** 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 4.** At the upper right of the **Alarms** page, click the **Table settings & actions** button.

**Step 5.** Select **Suppress** from the list.

**Step 6.** Optionally, you can choose to suppress the alarm only temporarily by doing either of the following:

- Click the drop-down control and click one of the standard periods displayed.
- Click the drop-down control, then click **Custom**, and in the resulting window select a date and time for the suppression to expire.

**Step 7.** Click **Confirm** to complete the suppression 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.

## 9.12 Viewing alarm details

### About this task

You can switch between the **Alarm Details** page and the **Alarm History** pages by selecting **Details** or **History** from the drop-down list at the upper right of the page.

The bread crumb path allows you to navigate from the **Alarms Summary** page to the **Alarms Details** or **History** page.

*Figure 50: Bread crumb path and drop-down menu*


*sc0265*

### Procedure

**Step 1.** In the EDA main menu, select **Alarms** to open the **Alarms** page.

**Step 2.** Find the alarm in the list using the sorting and filtering controls.

**Step 3.** Double-click the alarm.

**Expected outcome**

EDA opens the **Alarm Details** page, which shows the Alarm Summary information including the following grouped fields:

- Alarm Summary – includes Name, Namespace, Type, Severity, Cleared, Resource, Description, Kind, Group, Occurrences, Acknowledged, and Last Changed fields
- Remediation – includes Parent Alarms, Probable Cause, and Remedial Action fields
- Additional Properties – includes Acknowledged Until, Last Acknowledged, Suppressed, Suppressed Until, Last Suppressed, Source Resource, Source Kind, Source Group, and JS Path fields

Fields that are empty or not applicable are hidden.

## 9.13 Viewing alarm history

### About this task

You can switch between the **Alarm History** and the **Alarm Details** pages by selecting **Details** or **History** from the drop-down list at the upper right of the page.

The bread crumb path allows you to navigate from the **Alarms Summary** page to the **Alarms Details** or **History** page.

*Figure 51: Bread crumb path and drop-down list*


*sc0265*

### Procedure

**Step 1.** In the EDA main menu, select **Alarms** to open the **Alarms** page.

**Step 2.** Find the alarm in the list using the sorting and filtering controls.

**Step 3.** At the right side of the row, click the **Table row actions** button.

**Step 4.** Click **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 (true/false)
- Last change date and time
- Probable cause
- Remedial action
- Acknowledged (true/false)
- History of users who acknowledged

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

# 10 EDA query language (EQL)

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

**Note:** In addition to using the UI, EDA Query Language and Natural Query Language response can be extracted using API request with medias types of the following:

- application/json (default)
- application/yam
- text/csv

The media type application/xml is not currently supported.

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*

## 10.1 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 file is its own table: `.namespace.node` is a table, `.namespace.node.srl` is a table, and `.namespace.node.srl.interface` is a table.

A `Table` can be identified in the format as a complete JSPath format without keys. For example: `.namespace.node.srl.interface.subinterface`

### 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:

- `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 found value for a given field matching a `Filter`.
- `concat()` to merge multiple keys into a single field with a user defined delimiter.
  For example :

```
fields [ concat(.namespace.name, “/”, .namespace.node.name) as “Namespace/
Node”, .namesace.node.srl.interface.name as Interface ]
```

**Note:** `concat()` is used for EDA dashboards when a chart requires a single unique key (or a primary and secondary key), but the EDB path includes three or more keys.

### Filter

A Filter is a string defining any filters to use. A Filter is defined with a `where` term. The following rules apply:

- A `Filter` consists of an ordered set of fields, operators, values, and keywords.

  - Keywords may be capitalized or not. For example, both `and` and `AND` are valid.
  - Operators include:
    - Comparison operators with `where` clause - =, `!=`, <=, >=, >, <
    - `and`, `or`, and grouping constructs within `where` clause
    - `in` operator, allowing an array of values to be provided for comparison
    - `not in` operator, allowing an array of values to be provided for exclusion in the comparison
- 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:** Even when using a single `where` statement, it must be contained within `()`.

- EQL Filters support the `is set` and `is not set` operators within a `where` clause. Evaluations against an unset field will yield `True` for not equal and NULL for equal.
  - The evaluation follows the three-valued logic of `True`, `False`, and `Null` to ensure for logical consistency.
  - `not Null` is `Null`
  - `X and Null` is `Null`
  - `X or Null` is `X`
  - You can use `is set` and `is not set` to check for a field that is optional or has no default value. This comparison on a field checks whether the field has any value. If you want to include unset fields, you must also use `is not set` .
  - This can also be used to check for `Null` such as a case where `(mtu < 2000) is not set` since it behaves like unset field

### 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 control 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`.
- The third value is optional but currently can only be `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 (for example, 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`.

### Regular expressions with ~

Some EQL expressions need to match substring (or contains), prefix, and suffix matching on fields. To support these cases (among others), the ~ operator is supported. This allows the matching of regular expressions against values.

For example the expression `.field where (fieldname ~ "regex-pattern")` would match all objects in the `.field` table, where those objects have a field named `fieldname` and that field contains the string value `regex-pattern`.

The right side of the ~ operator must be a quoted string. If quotes are required inside the regex (for example, to match a literal quote character), you must escape them with a backslash (\).

The following regex operations are supported:

- Basic match, where prefixes and wildcards are allowed. For example:
  `.namespace.node.srl.interface where (name ~ "^ethernet-1/")`

  This expression would match all interface names that start with `ethernet-1/`. The `.*` at the end is implied. If that is not wanted, use a `$` to match the end of a value.
- Character classes, where one character is matched from a set. For example:
  `.namespace.node.srl.interface where (name ~ "^ethernet-1/[1,4]$")`

  This expression would match all interface names that start with `ethernet-1/`, then include either a `1` or `4`, with no other characters following.
- Alternation, where one of several alternatives is matched. For example:
  `.namespace.node.srl.interface where (name ~ "^ethernet-1/[1,4]$|^ethernet-1/20")`

  This is similar to the previous example, but would also match any interfaces the started with `ethernet-1/20`.
- The `!~` operator is not supported, but an expression may be wrapped in `not()` to perform the same function. For example:

```
.namespace.node.srl.interface where (not(name ~ "^ethernet-1/[\\d]$|
^ethernet-1/20"))
```

- All other typical regex operators are supported:
  - `\\d` to match a digit.
  - `\\w` to match an entire word (to the next whitespace).
  - `\\s` to match whitespace.
  - `\\n` to match a newline.
  - `.` for matching any character except a newline.
  - *, +, ? for matching the preceding element 0 or more, one or more, or zero or one respectively.
  - ^ to match the start of a string.
  - $ to match the end of a string.

## 10.2 Consolidated JSON to support EQL

It is common to query for values that are presented as a single object in the EDA GUI, but in fact span multiple containers or lists as they are normally stored within EDA. Such queries can exceed the usual supported scope of a single EQL query.

For example, a query like "show me all 100G interfaces that are enabled" requires data involving the `enabled` field which is stored as `.spec.enabled`, and the `speed` field stored as `.spec.ethernet.speed` .

To support such queries, Custom Resource (CR) entries are merged into single nested table entry so that:

- all configuration and state are in same table
- queries can thereby span all fields within an object

Queries against sub-tables of objects are not supported. Queries must start at the root of the object, with syntax resembling the following examples.

For the 100G speed and Enabled query described earlier:

```
.namespace.resources.cr.interfaces_eda_nokia_com.v1alpha1.interface where (spec.ethernet.speed
 = "100G" AND spec.enabled = "true")
```

To include "state" in the same query:

```
.namespace.resources.cr.interfaces_eda_nokia_com.v1alpha1.interface where (spec.ethernet.speed
 = "100G" AND spec.enabled = "true" AND .status.operationalState = "down")
```

## 10.3 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.

## 10.4 Creating a query with EQL

### Procedure

**Step 1.** Use the **Main** navigation panel to select **Queries** to open the **Query Builder** page.

**Step 2.** In the query types drop-down list, click **EQL Query**.

**Step 3.** 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 4.** When you have completed the query expression, click **Query** to view the results.

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

## 10.5 Creating a query with natural language

### Procedure

**Step 1.** Use the **Main** navigation panel to select **Queries** and open the **Query Builder** page.

**Step 2.** In the query types drop-down list, click **Natural Language Query**.

**Step 3.** 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 4.** 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.

# 11 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 configuration individually as you create it. EDA allows you to 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.

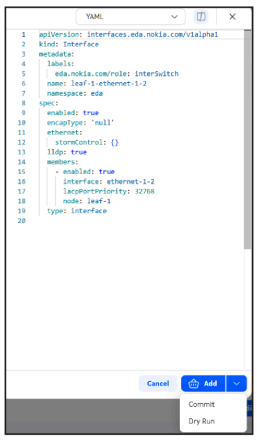
After a transaction has been successfully processed, it is written to EDA's Git repo for persistence, becoming the new accepted state of the infrastructure.

### Commit options for a typical resource

When you configure a resource in EDA, you are always presented with the following options to either:

- **Add**: add the change to a pending transaction for committing later
- **Commit**: commit the change by itself, immediately
- **Dry Run**: perform a dry run of the commit, to see what the results would look like if you were to proceed with committing this change

*Figure 52: A sample resource ready to commit*



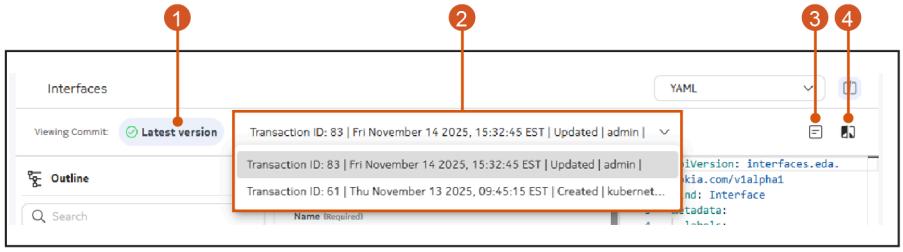

sc0236

## Transaction status bar

Every committed resource configuration is associated with a particular transaction. As you alter the resource over time, it accumulates a permanent commit history representing all the versions of itself that were ever committed as part of a transaction.

By default, any page in EDA that shows a configured resource shows its latest version. But you can view past versions of the same resource using the transaction status bar at the top of the page.

*Figure 53: Transaction status bar*




sc0309

*Table 35: Elements of the Transaction status bar*

| # | Name | Function |
|---|---|---|
| 1 | Version indicator | Shows whether the currently displayed configuration is the latest version, or some previous version. |
| 2 | Transaction selector | Displays a list of every transaction that includes a change to this resource's configuration. Select a past transaction to see a previous version of this resource.<br><br>**Note:** When you select a past transaction, the option to Revert is displayed as a button at the bottom of the page. Clicking **Revert** opens an editing window showing the |

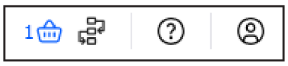
| # | Name | Function |
|---|---|---|
| | | modifications that reversion would make to the current version of the resource. You can then proceed with those changes, or alter them further. When the changes are correct, you can Dry Run, Commit, or Add the changes to the current transaction, just like any other change to a resource. |
| 3 | Navigate to Transaction button | Click to view details about the transaction currently show in the **Transaction selector** drop-down. |
| 4 | Diffs view button | Click to see a before-and-after comparison showing the specific changes that were applied by the resource as part of the selected transaction. |

## 11.1 Transactions drop-down panel

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

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

*Figure 54: The basket showing one pending resource*




sc0268

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

The **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 resource
- remove a resource from the transaction
- restore a removed resource back into the transaction

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

*Figure 55: The transaction drop-down panel*

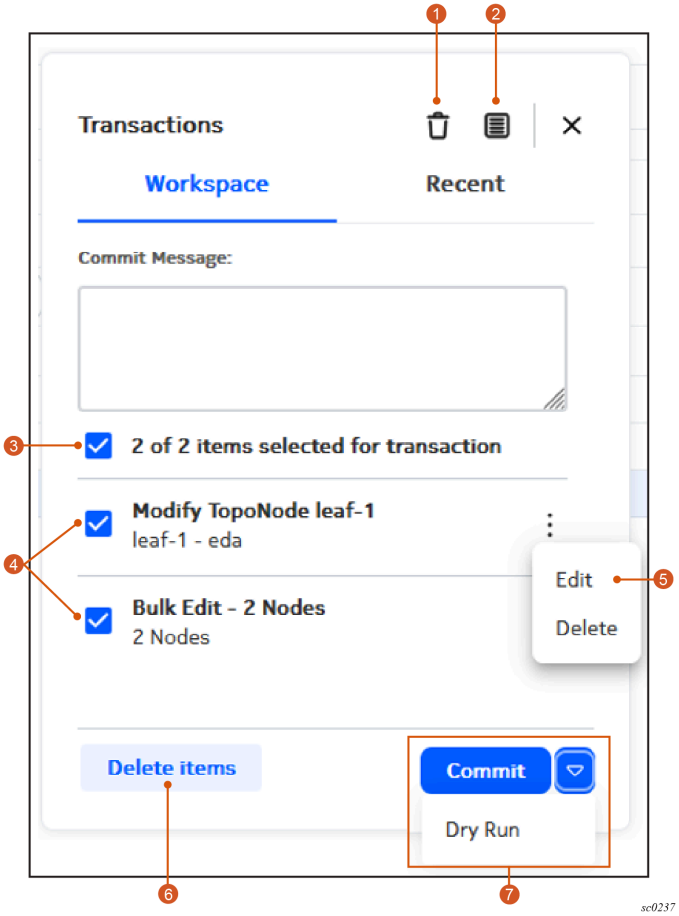



*Table 36: Elements of the Transactions drop-down*

| # | Description |
|---|---|
| 1 | **Clear Workspace** button: click to clear all configurations from the Transactions workspace. |
| 2 | **Log** button: click to open the Transactions log. |
| 3 | **Select All** button: use to select all configurations within this transaction. |
| 4 | **Configuration selectors**: Check or un-check to include a configuration within, or exclude a commit from, the current transaction.<br>Use the main check box at the top of the list to include or exclude all configurations together. |
| 5 | **Configuration Menu** button: use to access the Edit or Delete actions for this configuration. |

| # | Description |
|---|---|
| 6 | **Delete items** button: click to clear only the selected configurations from the transaction. |
| 7 | Transaction options:<br>• **Dry Run**: perform a dry run to evaluate the effects of the transaction after it is committed.<br>• **Commit**: proceed with applying the selected configurations as part of a single transaction. |

### Dry runs

In EDA, you can perform a dry run of any pending transaction. A dry run 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 resources.

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 resource, and validates the transaction against that data. The result of the dry run is displayed within the drop-down form.

During a dry run, you cannot make changes to the commit. After the dry run is complete, transaction details and diffs are available just as with a normal commit.

Closing the transaction basket dismisses the Transactions form, but retains information about the most recent dry run. If changes occur to the basket contents while the form is closed, dry run information is cleared and it is restored to the normal list view.

*Figure 56: A sample result of a dry run*

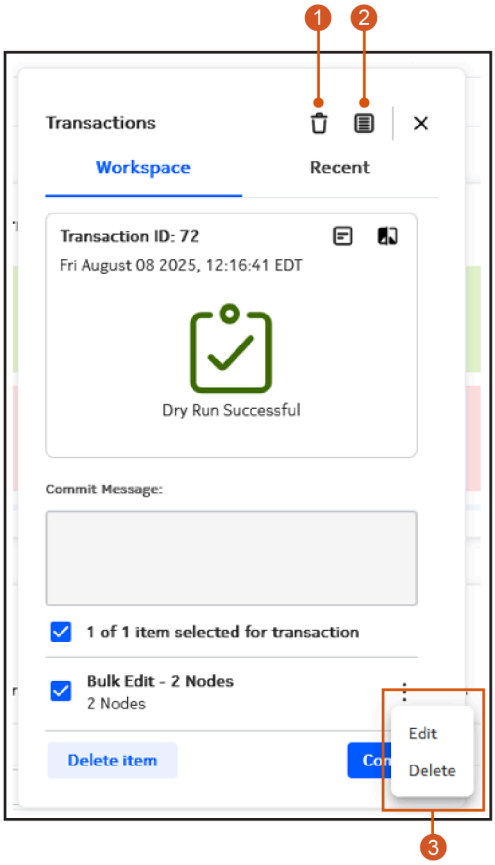



*Table 37: Elements of the dry run results display*

| # | Description |
|---|---|
| 1 | **Details** button: click to open the Details view for this transaction. |
| 2 | **Diffs** button: click to open the Diffs view for this transaction. |
| 3 | **Actions menu** for each row: use to edit or remove resource configurations within the transaction. |

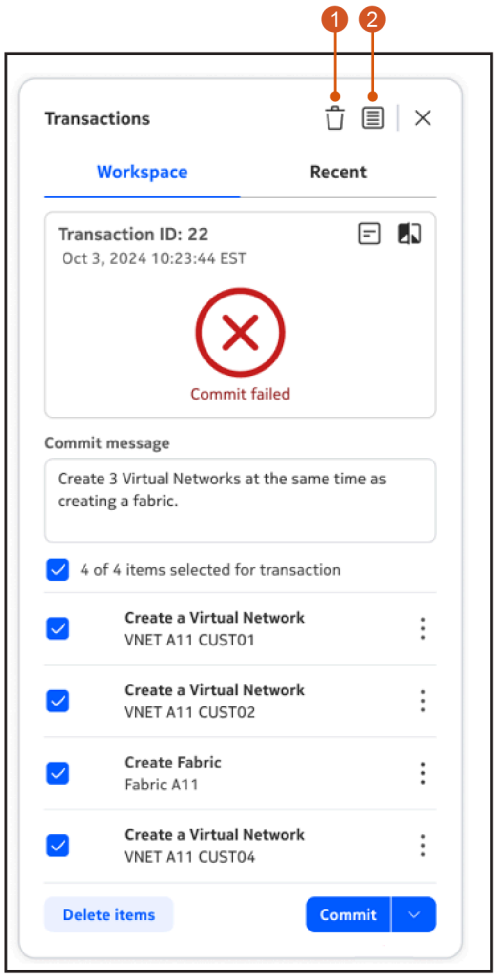
## Commit

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 commit the transaction again.

Whether the transaction succeeds or fails, you can view the details for the transaction, and the diffs that resulted from it. For a failed transaction, you can edit each row within the transaction to revise the configuration changes in support of another attempt.

Closing the transaction basket dismisses the Transactions form, but retains information about the most recent commit. If changes occur to the basket contents while the form is closed, commit information is cleared and it is restored to the normal list view.

*Figure 57: A sample of a failed transaction with one error*




*sc0267*

*Table 38: Elements of the transaction results display*

| # | Description |
|---|---|
| 1 | **Details** button: click to open the Details view for this transaction. |
| 2 | **Diffs** button: click to open the Diffs view for this transaction. |

## Recent

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

From the **Recent** view, you can view the details or diffs view for any listed transaction.

### 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: Un-check the **Configuration selector** check box beside any configuration to remove it from the overall transaction.
- **Restore** the configuration: Re-check the **Configuration selector** check box beside any unchecked configuration to restore it to the overall transaction.

### Additional actions

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

- **Discard Transaction**: empties the basket. This can be done before any commit/dry run, in which case nothing is in the transaction log.
- **Transaction log**: select this action to open the Transactions page, showing a list of recent transactions.

## 11.2 Transactions page

The Transactions log displays all of the transactions recorded in the Transactions log, providing an overview of transactions performed in the EDA system.

The Transaction log is visible to all authenticated users.

From this page you can:

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

**Note:** The Transactions log retains the complete details of past transactions only for the most recent 25 transaction for each user. All successfully committed transactions are still retained in the list list beyond that limit, but display only "Basic" details (see Transaction details). Dry-run and failed transactions are not retained after the 25 limit.

The following information displayed in the transactions log:

*Table 39: Transaction log displayed columns*

| Column | Description |
|---|---|
| ID (hidden by default) | A unique ID for the transaction assigned internally. |
| Transaction ID | An identifier for the transaction, in the form "Transaction <number>". |

| Column | Description |
|---|---|
| Transaction type | Whether the transaction was a Dry Run or commit.<br>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. |
| Status | Whether the transaction was successful, or failed. |
| Description | An optional description provided at the time the transaction is committed. |
| Created by | The user who executed the transaction. |
| Completion Timestamp | The date and time the transaction was completed. |
| State | The state of the completed transaction. |
| Commit Hash | A unique string that identifies this transaction in the EDA repository. |

### Transaction row actions

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

- **Details**, which opens the EDA **Transaction Summary** page showing detailed information for one transaction.
- **Diffs**, which opens the EDA **Transactions Diffs** page.
- **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.

  **Note:** Executing a Revert operation requires Read-Write permission for all input resources in the transaction.

- **Restore**: a powerful command that restores all resources to their state at the completion of the selected transaction. A confirmation dialog displays before the Restore operation proceeds.

  **Note:** Executing a Restore operation requires readWrite permission for the transaction restore API endpoint.

- **Add to Transaction**: adds the transaction's input resources to the transaction basket. This allows you re-try the transaction (or a modified version of a previous transaction).

### Reverting a transaction

When you choose to revert a transaction, EDA creates a new, additive commit that applies a set of changes that are opposite to those in the transaction being reverted. Committing this change has the effect of undoing the reverted transaction.

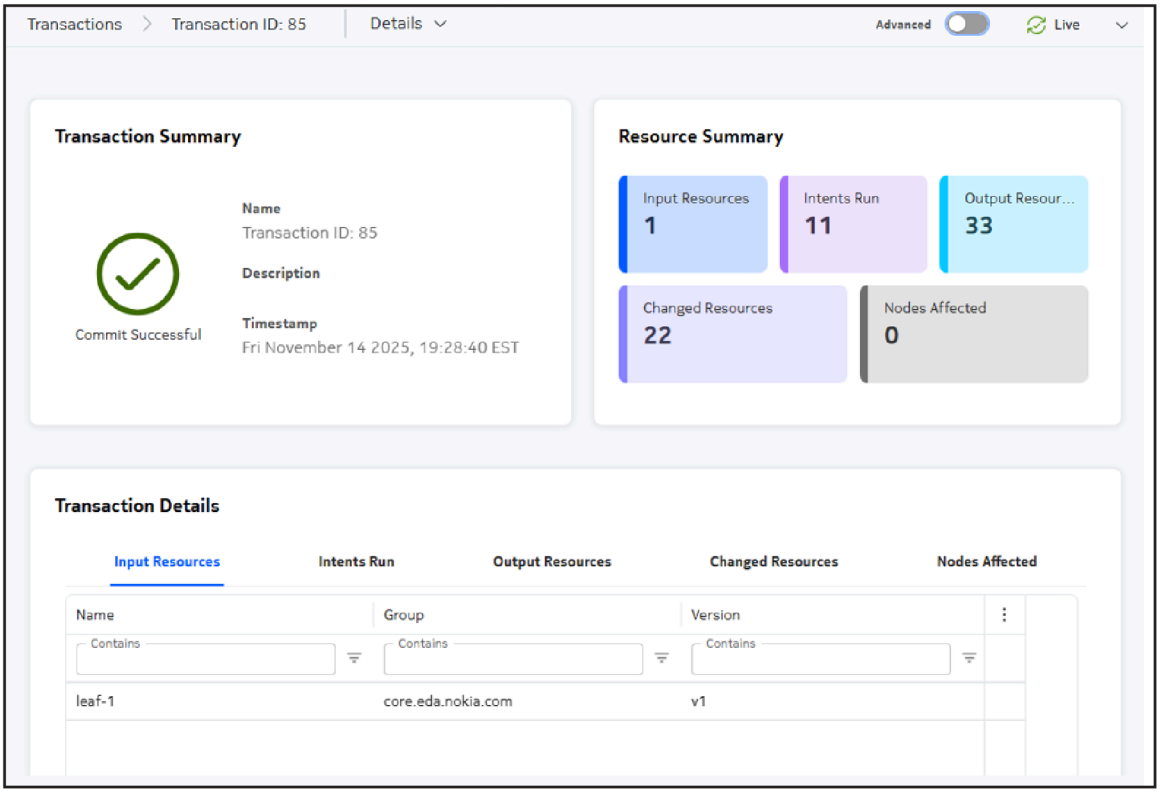
However, if there have been additional changes to the affected resources, subsequent to the transaction being reverted, this can produce unexpected results. To avoid this, EDA always causes such a revert action to fail.

To revert such a transaction, first revert any subsequent transactions that affected the same resources.

## Transactions summary

The **Transaction Summary** page shows detailed information about a single transaction.

*Figure 58: The Transactions summary page*




*sc0239*

This includes:

- The completion status, the name of the transaction, any description provided, and a timestamp for its completion.
- If there were errors, an Error Summary panel showing the number of:
  - General errors: errors related to the EDA environment
  - Intent errors: errors returned by an intent script
  - Node configuration errors: Errors in YANG schema validation, or errors returned by the node when pushing configuration
- A Resource Summary panel, showing the number of:
  - Input Resources: resources created, updated or deleted by the user
  - Intents Run: configuration scripts executed during the transaction)
  - Output resources: resources derived from the intent run
  - Changed Resources: input and output resources that are changed, compared to the previous committed transaction

  - Nodes Affected: nodes which are impacted by this transaction. This includes node configuration changes, node version changes, or changes to the associated TopoNode resource in EDA

**Note:** The Input Resources list includes only resources for which the user has read access.

**Note:** Intent runs, Changed Resources, Nodes with Changes, and Errors are only visible to users if they have read access to all Input Resources for the transaction.

### Transaction details

The errors and resources 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 resources, changed resources, intents run, nodes affected, and errors.

Some details of the transaction results, such as node configuration diffs, are stored in memory for a limited time.

Initially, all transactions include either a 'standard' or 'detailed' set of results. Transactions created via EDA UI always use 'detailed', whereas 'standard' is the default for transactions submitted via the EDA API or Kubernetes interfaces.

On clean-up, older committed transactions are reduced to the 'basic' detail level which includes only data stored in the Git repository. Failed and dry-run transactions are purged from this history on clean-up.

- Basic results include:
  - Input Resources
  - Input Resource diffs
- Standard results include all basic results, plus:
  - Errors
  - Intent Runs
  - Output Resources
  - Nodes Impacted
- Detailed results include all standard results, plus:
  - Changed Resources
  - Changed Resource diffs
  - Node Configuration Diffs

When only Basic or Standard results are available, a banner at the top of the **Transaction Details** page explains that this limitation is in effect.

EDA uses the following rules for reducing transaction result details:

- Keep details for each users 25 most recent transactions. Reduce details from any older transactions.
- When more than 10000 resource diffs are in memory, reduce details from the oldest transactions until there are less than 10000 resource diffs.

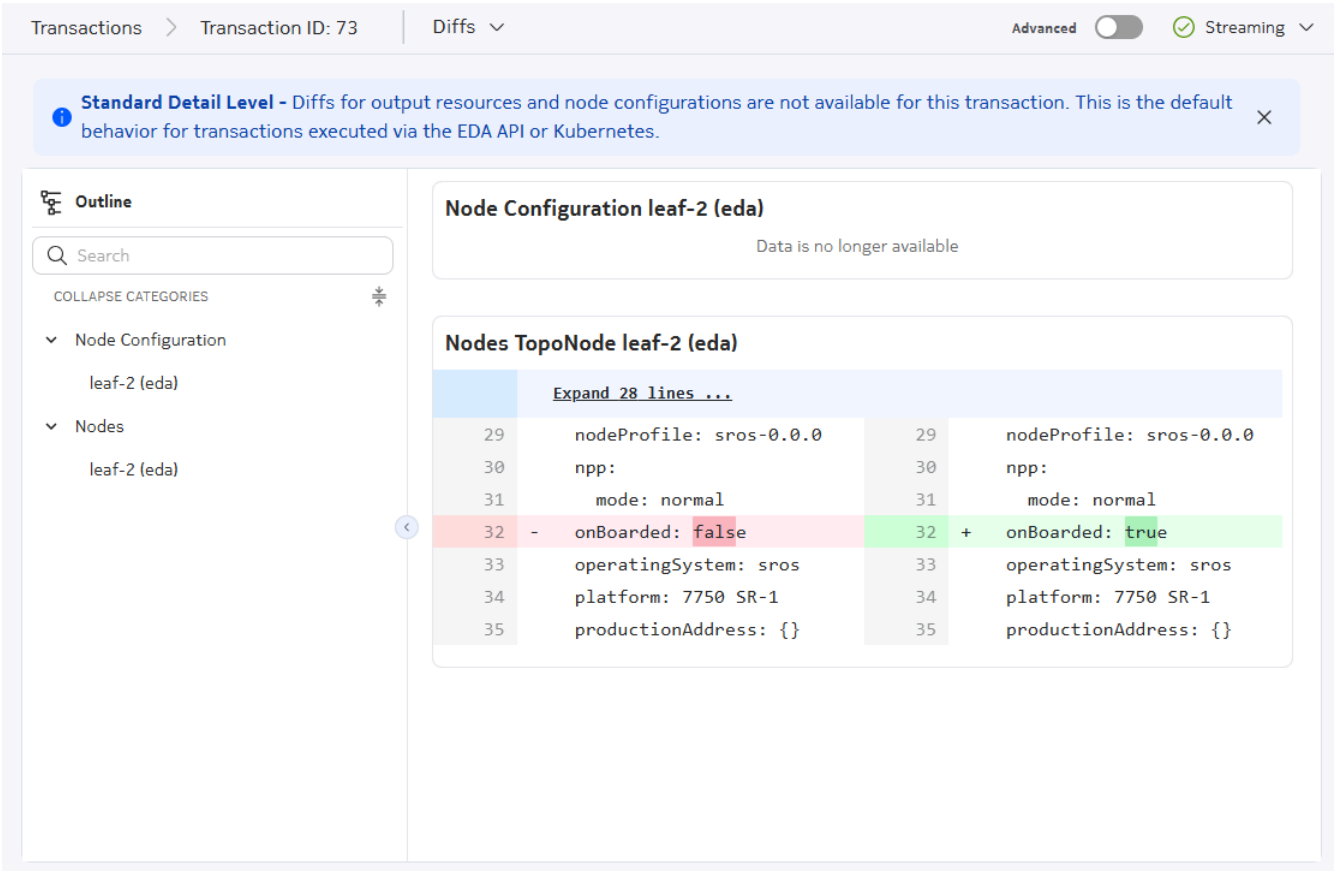
Transaction Output Resources often include internal EDA resource that are not user-facing. For example, an EDA configuration intent may create a "state" resource that initializes the process for reporting operational status.

You can use the **Advanced** toggle at the top of the **Details** page to control what level of detail is displayed regarding transactions:

## Transaction diffs

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

*Figure 59: The Transaction Diffs view*




Each changed CR and resulting node configuration change as part of this transaction is displayed, along with a diff view representing the new or changed lines in the respective configuration data.

**Note:** Resource diffs are only visible to users if they have read access to all Input CRs for the transaction.

**Note:** Node Configuration diffs are only visible to users if they have read access to the transaction nodeconfig diff API endpoint.

## Transaction topology

The **Transaction Topology** page displays all of the input resources and output resources that are involved in a transaction. It graphs the relationship between derived and parent resources.

Resources and node configurations are displayed as nodes within the topology.

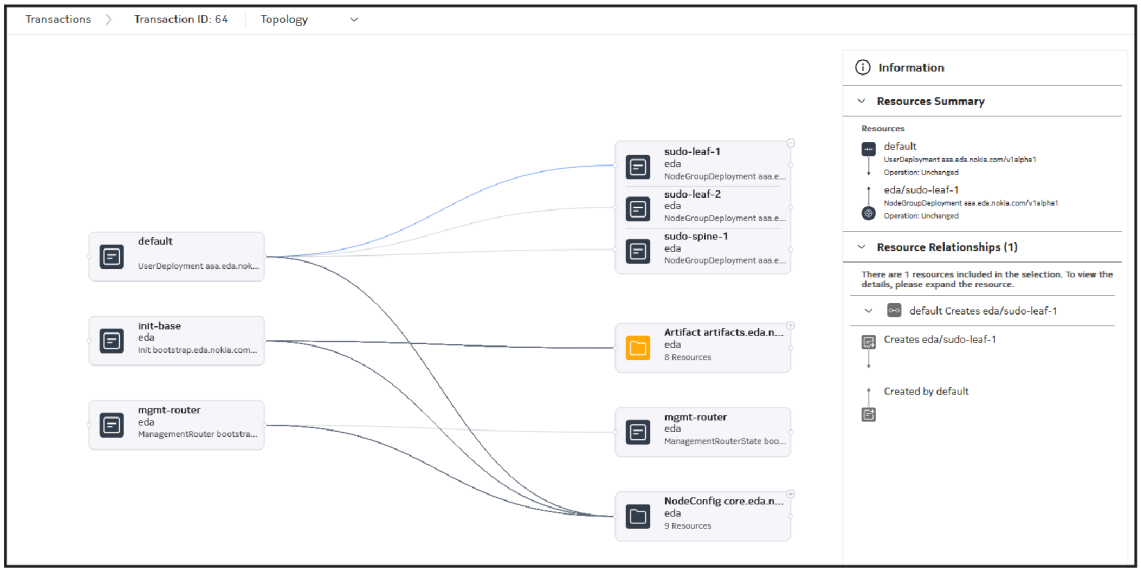
Links in the **Transaction Topology** page represents resource relationships established by the intent runs where:

- endpoint A represents the resource which triggered the intent script
- endpoint B represents the resource that was updated or created by the intent script

Changed resources are shaded yellow in the topology.

If an intent error occurred during the transaction, the resource that triggered the intent script is shaded red.

*Figure 60: The Transactions Topology view*




*sc0241*

**Note:** The **Transaction Topology** page only graphs createUpdate relationships. For example, when a fabric intent creates ISL resources, this is represented as links between the fabric resource and those ISL resources.
The **Transaction Topology** page does not graph Read relationships. So, for example, when a fabric intent reads information from allocation pool resources, the links between the fabric resource and allocation pool resources are not illustrated in the **Transaction Topology** page.

Some elements in the illustration represent a group of resources. Click the **+** at the upper right of such elements to expand the group and see the complete set of individual resources.

Additional information about any selected element or connection within the transaction topology is available from the **Information** panel.

## 11.3 Adding a resource configuration to a transaction

### Procedure

**Step 1.** Create any resource in the EDA GUI.

**Step 2.** Click **Add** 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*

## 11.4 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 resource is in the basket, the basket icon is highlighted and displays a count of the pending resource changes available to commit.

**Step 2.** Optionally, click the action drop-down list and 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 resource 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.

# 12 Node management

This chapter provides the following information:

- bootstrapping and related procedures
- background information about zero-touch provisioning (ZTP)
- how to display information about and interact with nodes using the following views from the **Nodes** section from the **Main** menu of the EDA UI:
  - **Resources** view
  - **Deleted Resources** view
  - **Discovered Resources** view

**Related topics**
*Topology*

## 12.1 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 nodes 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 systems. 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 HTTP/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.

### 12.1.1 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 the 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
NnAAAAAABgKWcAAAAADohaAAAAADAACQAoKr6XCCQCZj1rWFYik1dGbiqG7TWRK2orh
+0sjUKXNYBACkAMDAwMDAwMDAtMDAwMC0wMDAwLTAwMDAtMDAwMDAwMDAwMDAwAAAAAYADAABAAQAAAAAAMAAMAC/
KQqX7Di/m1d0zYz9quIyghaHatF0yDvDgK/fFr011Wa/7FN3LO/OoD3a
Hg8AXQFkOEh6ejQrTlFyNmVJNTNsVW9SMi9JV2xXd1NqMUF3QVh0eEd6LzhGdlp0WXph
TkdOQ1RWRnNCQ3wwZ0p2b21pSDNiZHFTSFBYQ2R6d0xxVlNhM3FZZUZuL1BGMnho
SjN6OS8yS3RlVGpmUngreWFNS1NwZ0p5OE12YlBVbmw2TUFpNHRXR1g4U3R0WXFBN21uVUNhVHp5e
XpLOWtXcWgwZVZtR1oyV09RTURML0thaWY1RGMva21tc0NVY042RUdNZUNi
TmdvV2RKUFlXZ1o4c2hlaG03b2tsZHdsSDBxMXZWdjhHMjZ4OVUxbTd2ellBN3BDNkFXODJyZ3FsaExWTUJx
Ym11VDdKSzdPWWhzYVp4Q3h4a2lIbWZ5KytNY3FLVHFBUk1McWhYRzRIb290ME0xK1RaRVZTdUJKNFl5a3pke
HdVV3pGZGRZdjg5Ym5uUHBsdXc9PQAAAAA="
```

- If the deployment uses EDA DHCP for bootstrapping, the `GlobalConfig` and `UdpProxy` CRs 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
---
apiVersion: artifacts.eda.nokia.com/v1
kind: Artifact
metadata:
  name: sros-iom-24-10-r4
  namespace: eda
spec:
  repo: images
  filePath: iom.tim
  remoteFileUrl:
    fileUrl: ftp://10.10.10.10/fsp/sros_images/24.10.r4/iom.tim
  secret: srl-ftp-cred
---
apiVersion: artifacts.eda.nokia.com/v1
kind: Artifact
metadata:
  name: sros-both-24-10-r4
  namespace: eda
spec:
  repo: images
  filePath: both.tim
  remoteFileUrl:
    fileUrl: ftp://10.10.10.10/fsp/sros_images/24.10.r4/both.tim
  secret: srl-ftp-cred
---
apiVersion: artifacts.eda.nokia.com/v1
kind: Artifact
metadata:
  name: sros-support-24-10-r4
  namespace: eda
spec:
  repo: images
  filePath: support.tim
  remoteFileUrl:
    fileUrl: ftp://10.10.10.10/fsp/sros_images/24.10.r4/support.tim
  secret: srl-ftp-cred
---
apiVersion: artifacts.eda.nokia.com/v1
kind: Artifact
metadata:
```

```
  name: sros-cpm-24-10-r4
  namespace: eda
spec:
  repo: images
  filePath: cpm.tim
  remoteFileUrl:
    fileUrl: ftp://10.10.10.10/fsp/sros_images/24.10.r4/cpm.tim
  secret: srl-ftp-cred
---
apiVersion: artifacts.eda.nokia.com/v1
kind: Artifact
metadata:
  name: sros-kernel-24-10-r4
  namespace: eda
spec:
  repo: images
  filePath: kernel.tim
  remoteFileUrl:
    fileUrl: ftp://10.10.10.10/fsp/sros_images/24.10.r4/kernel.tim
  secret: srl-ftp-cred
---
```

### 12.1.2 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
```

**Note:** If the `ipv4DHCP` or `ipv6DHCP` parameters are set to `true`, the settings are not reflected in the DHCP client-related config in BOF for SR OS.

### 12.1.3 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 40: 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 an 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 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::"
```

## 12.2 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 (IP addresses) 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 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:

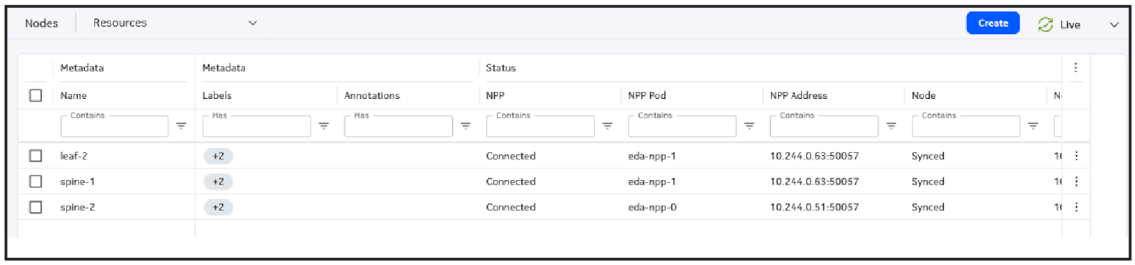
- Static lease assignment using the client-id (option 61) as the binding between an IP address and a device
- Ability to receive DHCP packets from a DHCP relay (the DHCP relay between the devices and the DHCP server)
- When providing an IP address to the device, the DHCP server must be able to populate 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

## 12.3 Node Resources page

The **Node Resources** view shows all the nodes known to the EDA system that are associated with a fabric.

*Figure 61: Nodes Resources view*




*sc0279*

The following table lists the default columns shown in the **Node Resources** page.

*Table 41: Default Nodes Resources display columns*

| Column | Description |
|---|---|
| Name | The name of the resource. |
| Namespace | The namespace to which this node belongs. |
| Labels | The labels assigned to this node. |
| Annotations | The annotations assigned to this node. |
| NPP | The current state of the connection between ConfigEngine and NPP. |
| NPP Pod | The NPP pod name. |
| NPP Address | The NPP address and port for this TopoNode. |
| Node | The current state of the connection between NPP and the node, which can be one of the following:<br>• TryingToConnect: NPP is attempting to establish connectivity to the node. |

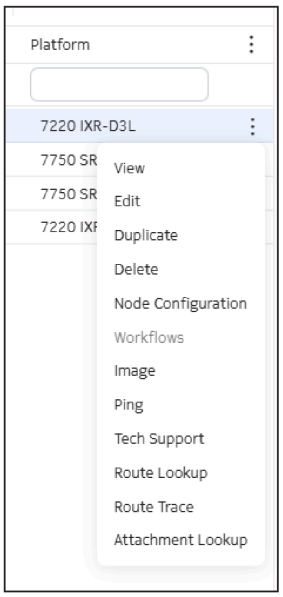
| Column | Description |
|---|---|
| | • WaitingForInitialCfg: NPP is connected to the node but is waiting for the initial config to push.<br>• Committing: NPP is in the process of committing.<br>• RetryingCommit: NPP lost sync to the node and is re-pushing current config.<br>• Synced : NPP is in a fully synched state.<br>• Standby: On geo-redundant clusters, NPP is running in standby mode.<br>• NoIpAddress: NPP is running, but the node has no IP address; occurs only in simulator setups when CX has not created the simulated node or the simulated pod failed to launch because of an image error. |
| Node Address | The address and port used to connect to the node. |
| Platform | The operational platform type of this node. |
| Version | The software version of this node. |
| Onboarded | Indicates if the node has been bootstrapped or is reachable using the configured credentials. |
| Operating System | The operating system running on this node. |
| MAC Address | The MAC address associated with this node. |
| Serial Number | The serial number of this node. |
| System Interface | Deprecated - no longer used. |
| License | The reference to a ConfigMap containing a license for the TopoNode. |
| Components | Details about the hardware:<br>• Kind: identifies the component, for example, if it is a line card.<br>• Type: the type of hardware provisioned.<br>• Slot: the slot in which this component resides. 1 indicates the line card in slot 1, 1/1 indicates line card slot 1, mda slot 1. |
| Mode | The mode in which this node is functioning:<br>• `Normal`: (the default) indicates that NPP is expecting an endpoint to exist and accepts and confirm changes only if the endpoint accepts them.<br>• `maintenance`: no changes accepted for the TopoNode, regardless of whether the endpoint is up and reachable, except if an upgrade is occurring, in which case changes are accepted.<br>• `null`: changes are accepted from CRs and no NPP is spun up. NPP validation does not occur.<br>• `emulate`: changes are accepted at the NPP level, without pushing them to a endpoint. NPP validation still occurs. Also displayed if no IP address is present. |
| Node Profile | The node profile applied to this node. |

| Column | Description |
|---|---|
| Production address: | The production addresses that this TopoNode uses. |

### Working with a node

Click the **Row actions menu** for a node to display the actions that you can perform on the node.

*Figure 62: Row actions*




sc0280

- **View**: display details about a selected node, including status, mode of operation, and so forth. You can also double-click a node from the list to display the details for a node.
- **Edit:** edit the settings of a node
- **Duplicate**: create a new TopoNode based on an existing node
- **Delete**: delete a node
- **Node Configuration**: view the node configuration (its CR)
- Under the **Workflows** options, you can run the following tasks with the node as the target:
  - update the image of the node
  - ping the node
  - create a help package for tech support
  - do a route lookup
  - do a route trace
  - do an attachment lookup

  For more information about these workflows, see Workflow Definition List page.

## 12.4 Creating a TopoNode

### Procedure

**Step 1.** Create a new node from the **Node Resource** view using one of the following options.

- Create a new node by copying the settings of an existing node. Locate the node and click the **Table row actions** icon at the end of its row. Select **Duplicate**. In the form that displays, some fields are pre-populated, but you need to provide node-specific information, such as the name, MAC address, serial number, license, and production addresses for the new node.
- Click **Create**. In the form that displays, the fields are all blank.

**Step 2.** Configure the metadata for this node.

Provide the following information:

- Name of the node.
- Labels for the node.
- Annotations for the node.

**Step 3.** Enter the specifications for the TopoNode. If you duplicated a node, verify the pre-populated settings and modify as needed.

Provide the following information:

- Platform
- Version
- Onboarded
- Operating System
- Node Profile
- MAC Address
- Serial Number (deprecated)
- System Interface
- License

**Step 4.** In the **Components** section, click **+ Add**.

In the form that opens, provide the following information for the node you are provisioning:

- the kind of component
- the type
- the slot in which this component resides

**Step 5.** Set the production address for this node.

These settings are required if the TopoNode is not bootstrapped by EDA. If left blank, an address is allocated from the management IP pool specified in the referenced NodeProfile resource. Click the **Specification | Production Address** toggle to enter the IPv4 and IPv6 production addresses for this node.

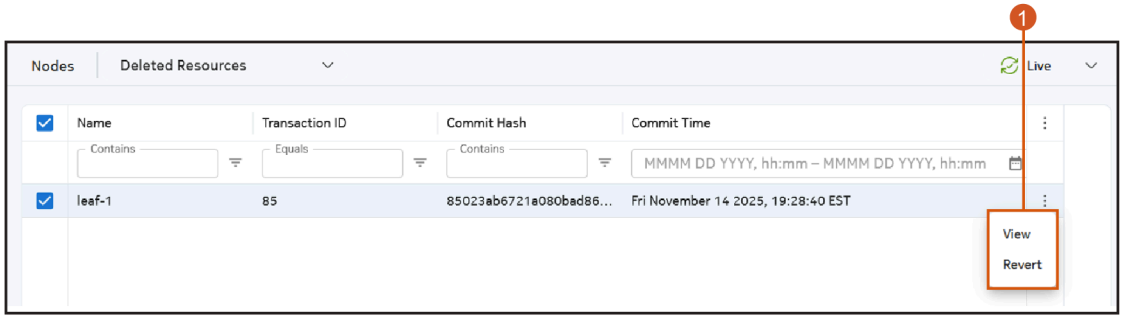
**Step 6.** Specify the operating mode of the node.

The default setting is normal.

## 12.5 Deleted Resources page

The **Deleted Resources** view displays all of the nodes that have previously been deleted from the system. From here you can view details about the reverted nodes, and if necessary, revert the deletion.

*Figure 63: The Deleted Resources view*




sc0281

*Table 42: Elements of the Deleted Resources view*

| # | Name | Function |
|---|---|---|
| 1 | Actions menu | Use the Actions menu to perform the following actions on a selected node:<br>• **View**: Opens the **Detail** view for the deleted node, clearly marked as a deleted version. Buttons on this view allow you to **Cancel** (close the **Detail** view) or **Revert** (to open the **Revert** view for this deleted node).<br>• **Revert**: Opens the **Detail** view for the deleted node, clearly marked as a deleted version. Buttons on this view allow you to **Revert** (to immediately restore this configuration of the deleted node) or **Add to Transaction** (to add the reversion to a transaction to be committed later). |

The list of nodes in the **Deleted Resources** view displays the following columns by default.

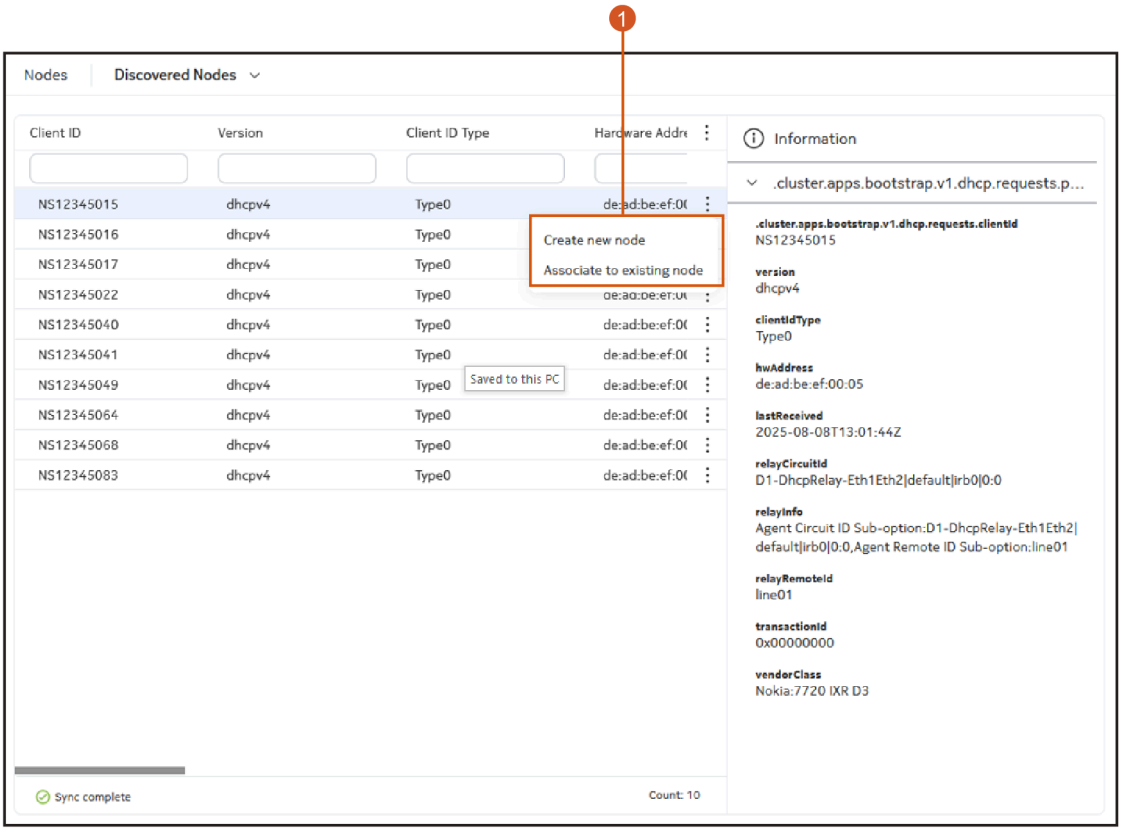
*Table 43: Default Deleted Resources display columns*

| Column | Description |
|---|---|
| Name | The name of the deleted node. |
| Namespace | The namespace to which the deleted node belonged. |
| Transaction ID | The ID of the transaction within EDA that deleted this node. |
| Commit Hash | The commit hash for the commit within EDA that deleted this node. |
| Commit Time | The date and time at which the deletion of this node was committed. |

## 12.6 Discovered Nodes page

The **Discovered Nodes** page contains a list of discovered nodes that have not been assigned to a fabric. When a discovered node is assigned to a fabric, it is removed from the Discovered Nodes list and added to the main list of nodes.

*Figure 64: The Discovered Nodes view*




sc0282

*Table 44: Elements of the Discovered Nodes view*

| # | Name | Function |
|---|---|---|
| 1 | Actions menu | Use the Actions menu to perform the following actions on a selected node:<br>• **Create new node**: Allows you to create a node from scratch or from an existing node.<br>• **Associate to existing node**: Allows you to copy relevant discovered node fields into a selected existing node. |

The list of nodes in the **Discovered Nodes** view displays the following columns by default.

*Table 45: Default Discovered Nodes display columns*

| Column | Description |
|---|---|
| Client ID | Indicates the ID sent by the client. The Client ID may refer to serial numbers, MAC addresses, or any other arbitrary string. |
| Version | Indicates the software version of the selected node. |
| Client ID Type | Indicates a string provided by the client. |
| Hardware Address | The MAC address of the client. |
| IP Address | The IP address of the client. |
| Last Received | Indicates the time at which a packet was last received by the client. |
| Namespace | Indicates the namespace to which the node belongs. |
| Node Name | Indicates the name of the node. |
| Relay Circuit ID | The parsed version of the Relay Info, indicating only the Circuit ID value, if any. |
| Relay Info | A raw dump of options passed by the DHCP, if any. |
| Relay Remote ID | The parsed version of the Relay Info, indicating only the Remote ID value, if any. |
| Transaction ID | The DHCP protocol specific transaction ID. |
| Vendor Class | The vendor class of the client. |

### 12.6.1 Creating a new node

#### About this task

You can create a new node from scratch or from an existing node.

#### Procedure

**Step 1.** At the right side of the Discovered Nodes list, click the **Table row actions** button.

**Step 2.** Click **Create a new node**.

**Expected outcome**

EDA opens the **Create a new node** page. To create a new node from scratch, go to step 3. To create a new node from an existing node, go to step 4.

**Step 3.** To create a node from scratch:

**a.** In the **Create a new node** page, click **Skip**.

**Expected outcome**

A blank form opens where you can create a new node from scratch.

**b.** Fill out the following required fields:

- Name
- Namespace

- Platform
- Version
- Operating System
- Node Profile

**c.** Click **Commit** to commit your change immediately or click **Add To Transaction** to add this item to transactions to commit later.

**Step 4.** To create a node from an existing node:

**a.** In the **Create a new node** page, select a single existing node from the list.

**b.** Click **Next**.

**Expected outcome**

A pre-filled form opens where relevant values are taken from the existing node. The following fields are pre-filled:

- Namespace
- Labels
- Platform
- Version
- Operating System
- Node Profile

**c.** Fill out any remaining required fields.

**d.** Click **Commit** to commit your change immediately or click **Add To Transaction** to add this item to transactions to commit later.

## 12.6.2 Associating to an existing node

### About this task

You can associate a selected discovered node to an existing node.

### Procedure

**Step 1.** At the right side of the Discovered Nodes list, click the **Table row actions** button.

**Step 2.** Click **Associate to existing node**.

**Expected outcome**

EDA opens the **Associate to existing node** page where you can select an available existing node to associate the discovered node to.

**Step 3.** Click **Next**.

**Expected outcome**

After a discovered node has been associated with an existing node, the discovered node is removed from the Discovered Nodes list.

# 13 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.

## 13.1 The Topologies page

You can view the topology page by selecting **Topologies** in EDA's Main navigation panel.

The page includes the following elements:

- Topology list: Displays a list of available topologies. Currently, the Physical topology is supported.

- The table row actions list: Select to take an action for the current row. Currently the **Show Topology** action is supported.
- Information panel: Like most pages 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.

By default, the topology displays in a horizontal, left-to-right orientation. You can use the orientation selector button to switch between a horizontal and vertical orientation for the illustration.

*Figure 65: The topology illustration*

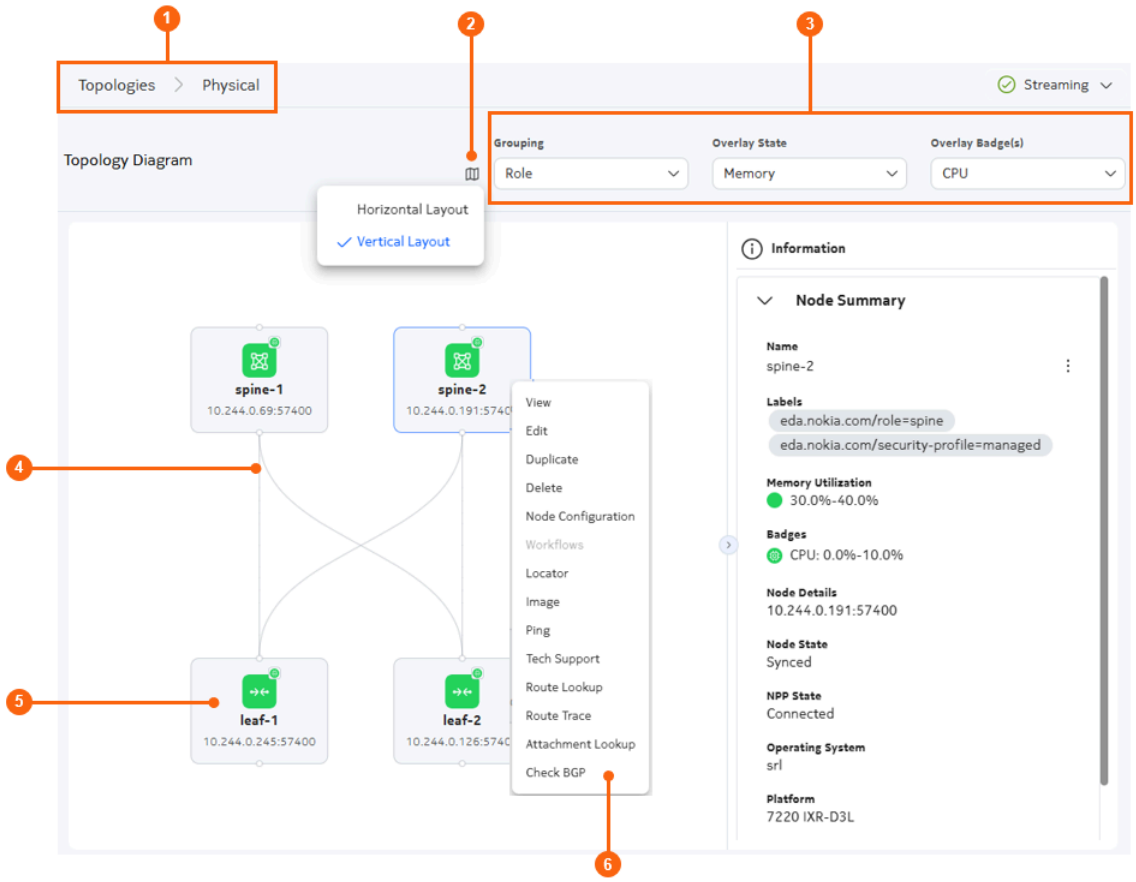



*Table 46: Elements of the topology illustration*

| # | Name | Function |
|---|---|---|
| 1 | Topology breadcrumb | Displays the name of the topology currently being displayed. You can click **Topologies** to return to the Topologies page. |

| # | Name | Function |
|---|---|---|
| 2 | Orientation selector | Click this button to select between a vertical (top to bottom) or horizontal (left to right) display for the topology elements. |
| 3 | Grouping, Overlay, and Badge selectors | Use this drop-down list to:<br>• Role: control the grouping of elements within the topology display<br>• Overlay: add shading to nodes to indicate the selected status<br>• Badges: add one or more badges to nodes to indicate the selected status.<br>Each overlay applies shading to the topology illustration to indicate the related status of all nodes and links.<br>Badges display on the nodes within the topology, and the badge icon and color indicate the related status of each node. You can select up to three badges for inclusion in the Topology display. |
| 4 | Links | These are examples of links within the topology.<br>Click on any link to view details about that link, the connected nodes, and the link endpoints in the **Information** panel. |
| 5 | Node | This is an example of a node within the topology.<br>Click on any node to view details about that node in the **Information** panel.<br>Right-click on any node to access the standard node actions menu. |
| 6 | Actions menu for nodes | Right-clicking on a node opens a menu for standard node actions. See the documentation for node management for details about these actions. |

## Grouping

The Grouping selection determines how elements within the topology are arranged within the topology diagram.

The Physical Topology includes one grouping, named **Role**, by default. To customize the physical topology grouping, create or edit TopologyGrouping resources via Kubernetes.

When grouped by **Role**, nodes are tiered from left to right based on the following labels:

- eda.nokia.com/role=backbone
- eda.nokia.com/role=superspine
- eda.nokia.com/role=leaf
- eda.nokia.com/role=borderleaf

Grouping can also control how multiple nodes are consolidated into a single expandable node within the topology illustration.

## Overlays

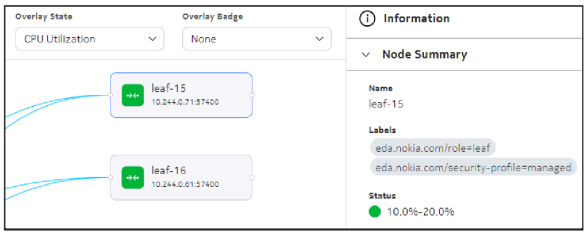
Information can be overlaid on a topology as either a state or as a badge. States shade the nodes and links of a topology with a color corresponding to the information. For example, EDA's Operational Status overlay shades nodes and link green for operationally up, and red for operationally down.

In EDA, you can toggle overlays on and off.

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

Overlays may also include state information for topology endpoints. This is not represented in the graph, but is available in the **Information** panel by selecting the appropriate link.

*Figure 66: CPU Utilization Overlay and information panel*




sc0244

The following overlays are available for physical topologies in EDA:

- The CPU utilization overlay shades nodes based on the percentage of CPU capacity that is in use on the node.
- The Memory overlay shades nodes based on the percentage of memory that is in use on the node.
- The Volume overlay shades nodes based on the percentage of disk usage on the node.
- The LLDP overlay shades links based on the discovered peer or interface on each endpoint, and the correlation of this with each link, to ensure alignment. It thereby acts as a kind of topology intent check.
- The Operational Status overlay indicates the operational status of each TopoNode and TopoLink member 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.

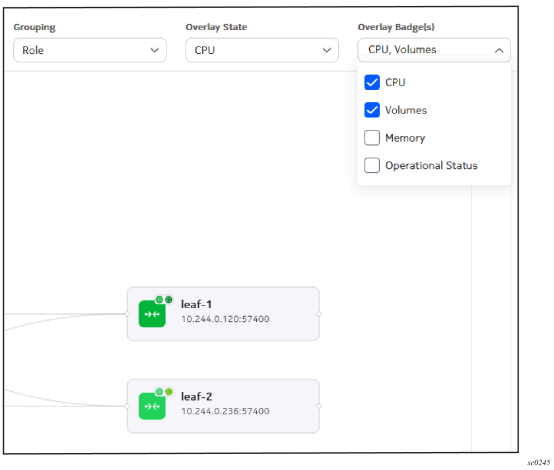
## Overlay badges

Overlay badges are displayed only on nodes within the topology. You can select up to three badges to display simultaneously. 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.

The Operational Status badge indicates the operational status of each TopoNode in the topology. This value is derived from the status of NPP connectivity to the node.

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

*Figure 67: A badge indicating that a node's Operational Status is 'Synced'*




## 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.

For links, the information panel includes nested summaries for each selected link showing its endpoints. Actions are available through a contextual menu associated with each eligible object.

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

*Figure 68: A sample information panel for a node, showing a contextual action menu*

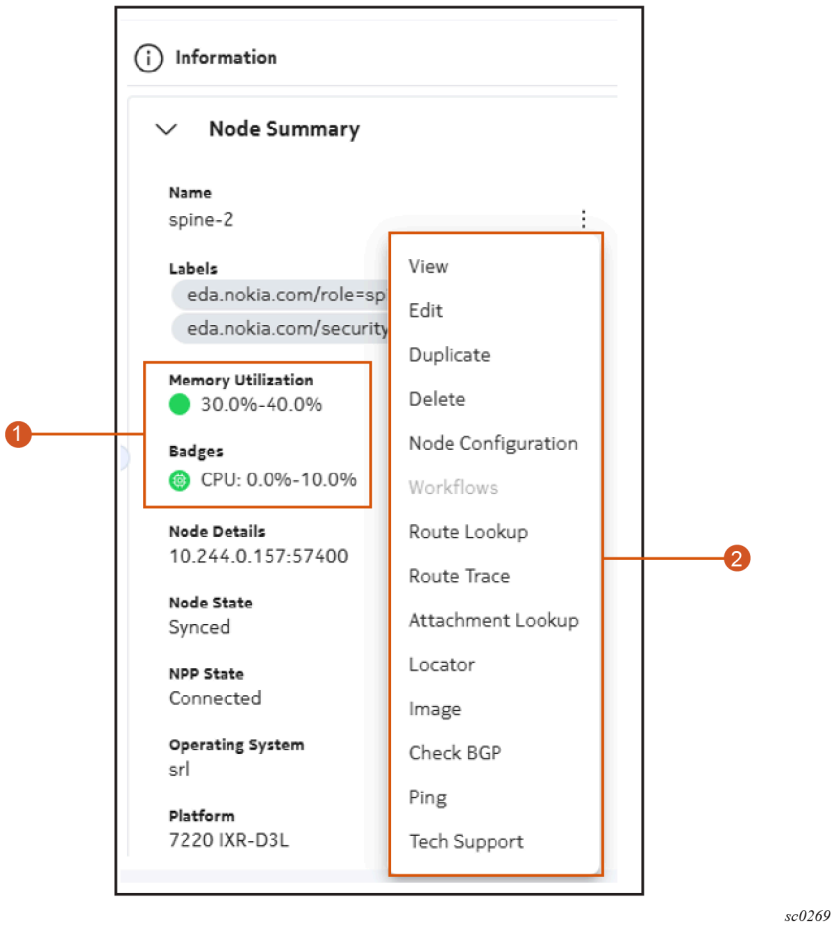
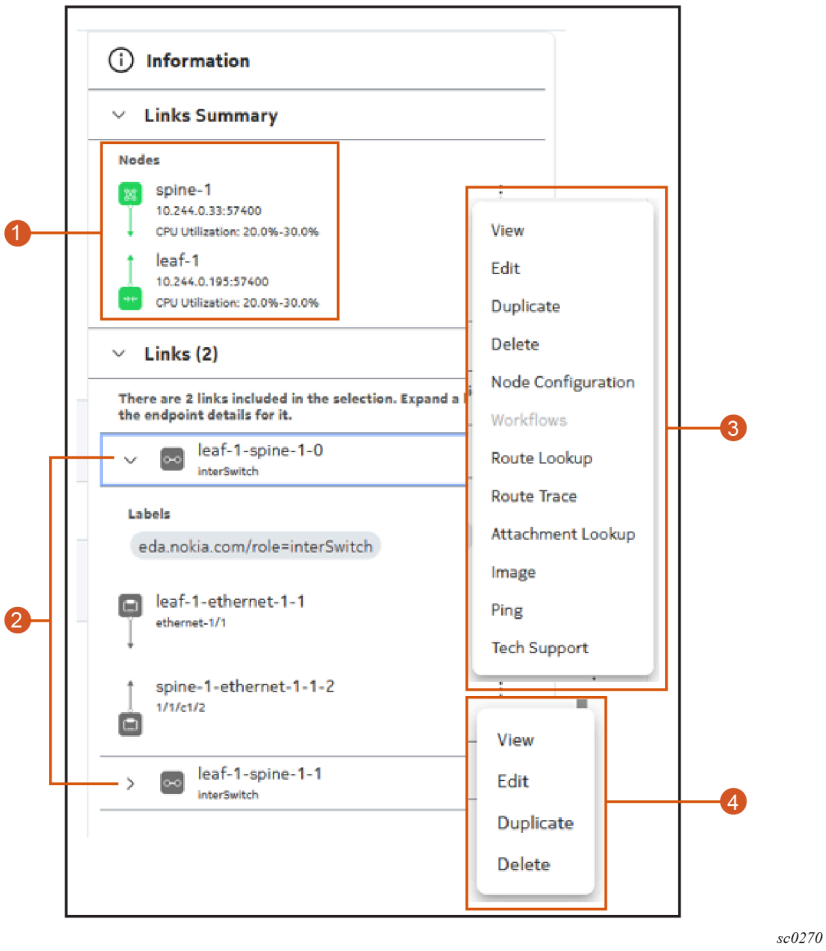



*Table 47: Elements of the node information panel*

| # | Name | Function |
|---|---|---|
| 1 | Status indicators | If an overlay or badges are enabled, then the associated overlay or badge names are displayed here along with the current, associated status for the selected nodes. |
| 2 | Actions menu | The conventional menu for node actions is available from the information panel. Actions are applied to the selected node. See the documentation for node management for details about these actions. |

*Figure 69: A sample information panel for a link, showing a contextual action menu*




sc0270

*Table 48: Elements of the link information panel*

| # | Name | Function |
|---|---|---|
| 1 | Participating nodes | Nodes at either end of the selected link are identified here. If an overlay is selected, that information is included here as well. |
| 2 | Links | The selected links are listed here. Initially they are collapsed. Expanding any link in the list displays details about its endpoint. |
| 3 | Node actions | A contextual action menu is available for displayed nodes |
| 4 | Link and endpoint actions | A contextual action menu is available for individual links (shown in the illustration) and endpoints (not shown). |

## 13.2 Viewing a topology

### Procedure

**Step 1.** Use the **Main** navigation panel to select **Topologies** and open the **Topologies** page.

**Step 2.** Click on a topology in the list.

**Step 3.** Use the **Table row actions** drop-down list to click **Show Topology**.

**Expected outcome**

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

**Step 4.** Optionally, select an overlay by clicking the **Overlay State** drop-down and selecting an overlay from the list.

**Expected outcome**

Nodes in the topology illustration are shaded to signify their status with regard to the selected overlay data.

**Step 5.** Optionally, select one or more badge types by clicking the **Overlay Badge** drop-down and selecting up to three badge types from the list.

**Expected outcome**

Nodes in the topology illustration display corresponding badges, shaded to signify the nodes' status with respect to the selected badge data.

**Step 6.** To view details about an individual node, select the node and then expand the **Information** panel.

The panel shows:

- identifying information for the selected node
- overlay and badge status for the selected node
- an action menu that includes supported node actions

**Step 7.** 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.
- an action menu for the nodes, links, and endpoints within the information panel

## 13.3 Resource topologies

Because the volume of resources and their relationships within EDA is very large, it can be difficult to effectively grasp the relationship between one resources, and all of the other configured resources on which it somehow depends.

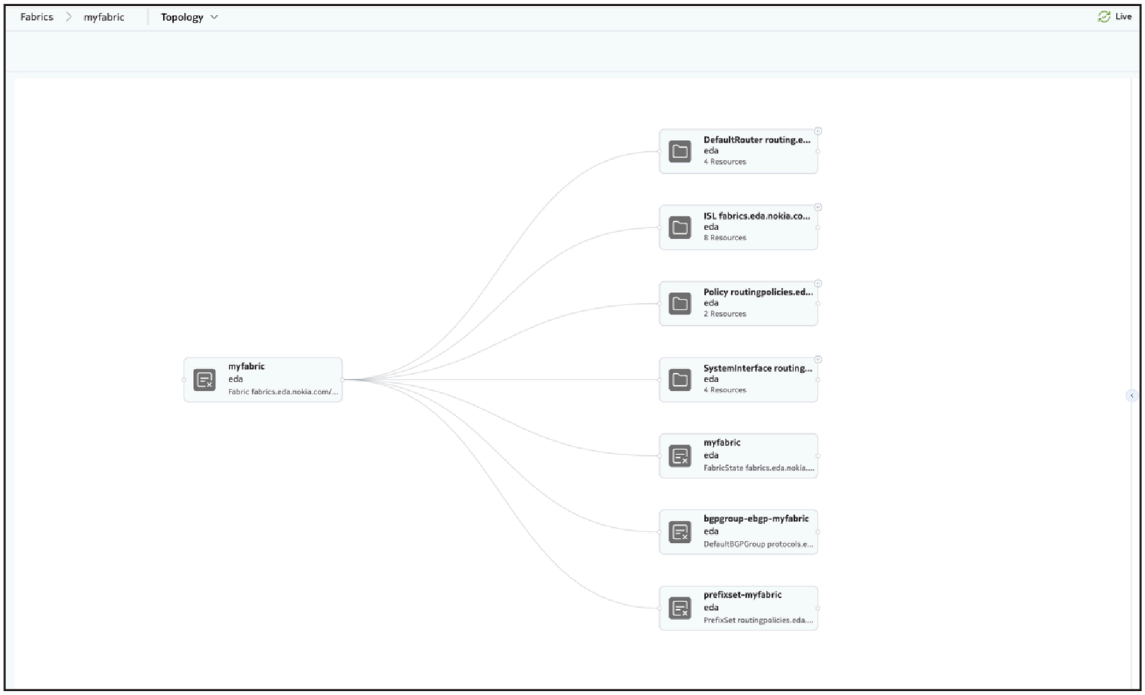
To help represent resources and their interconnections, EDA builds on its topology visualization framework by providing a Topology illustration. This illustration shows the selected resource, and the other EDA resources to which it is connected.

To see the topology view for a resource, open the **Details** view for an individual resource, and then select **Topology** from the drop-down list of available views.

For example, the following illustration shows the resource topology for a fabric. It shows not the fabric's physical topology, but its connection to the set of other resources configured within EDA:

- default routers
- ISLs
- routing policies
- system interfaces
- BGP groups
- prefix sets

*Figure 70: An example of a fabric resource topology*




*sc0310*

In the EDA UI, you can click any resource in the illustration to see more information in the Information panel.

# 14 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.

## 14.1 EDA Store

The EDA 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, you can access the **EDA Store** page from the **System Administration** navigation panel and selecting **Store**. EDA store displays all apps known to EDA in tile view. The tiles provides high-level details for each app.

If an upgrade is available, the tile shows an **Updates Available** button that opens the app page from where you can easily perform the upgrade.

Click an app to display app details. See Apps for more information.

You can sort, filter, and search for apps in EDA store.

### Filtering for apps

You can use the following fields to filter for apps:

- From the **Type** drop-down list, select from the following options:
  - **All** : displays all app packages know to EDA. This setting is the default when you open the **App Store** page.
  - **Updates Available**: displays apps for which an upgrade is available.
  - **Installed**: displays all the installed apps in the current EDA deployment.
- To filter for apps by catalog, click the **Catalog** drop-down list.
- To display apps by category, click the **Category** drop-down list. You can select all categories of apps, or core, demo, monitoring, networking, or system categories.
- To display apps by vendor, click the **Vendor** drop-down list.

### Sorting apps

To sort the apps alphabetically, click the **Sort by** drop-down list. You can sort from a to z and z to a.

## 14.2 App management

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

| | |
|---|---|
| **Catalog** | A catalog is a Git repository that contains the manifests of apps. |

| | |
|---|---|
| **Registry** | An EDA 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 EDA Store it must install an app in the EDA environment. |

## 14.3 Apps

### App manifest

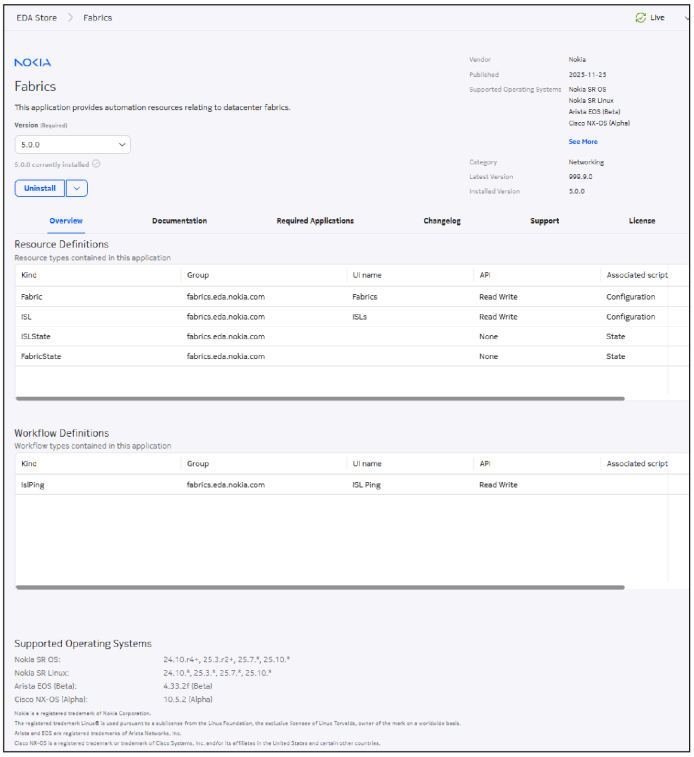
Apps are defined by a manifest that contains the following information about the app:

- 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
- Documentation link
- Artifacts
- Containers
- Scripts
- CRs
- CRDs
- UI design and configuration
- Dependencies and requirements

### App details

From the EDA Store, click any app to display details for that app.

*Figure 71: Example: Details for the Fabric app*




sc0307

This view provides the following information about the app:

- description of the app
- version of the app: to view other versions of the app, click the drop-down list
- vendor
- publish date
- supported operating system: click **See More** to display all supported operating systems. To view the supported versions for each operating system, see the **Supported Operating Systems** section in the **Overview** tab.
- category
- latest version
- installed version, if the app is installed

If the app is not installed, the **Install** drop-down list is displayed; see Installing an application for more information.

If the app is installed, the **Uninstall** and **Upgrade** options are displayed. For more information, Uninstalling an app.

The tabs shown in this view vary depending on the app.

- **Overview**: shows the Resource definitions and Workflow definitions that come with the app and details about the operating system supported.

- **Documentation**: provides documentation for an app.
- **Required Applications**: lists the required apps and if they are already installed; see App dependencies for details.
- **Changelog**: provides release notes for some apps.
- **Support**: for some apps, provides links to where you can get more information.
- **Licenses**: provides licensing information for the app.
- **Available settings**: displays only for specific apps; see App settings for details.

### App CRDs

The **Overview** tab lists the Resource and Workflow custom resource definitions (CRDs) that come with an app. The **Overview** tab provides the following information about resource and workflow CRDs:

- Kind / group
- UI name
- API (none, read only, or readWrite)
- Associated script type (none, config, or state)

### App dependencies

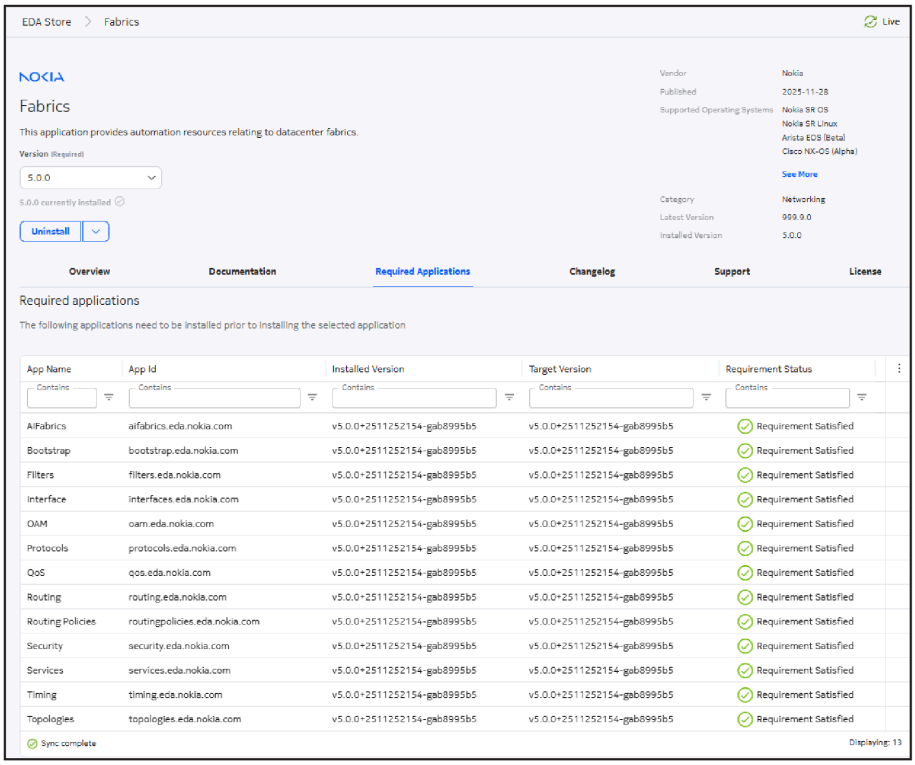
Dependencies are requirements that are defined in an app's manifest that must be satisfied before the app can be installed. The requirements can be for other apps or for a specified version of the EDA core.

If a dependency is not satisfied, such as when a required app is missing or if it is the wrong version, the impact is as follows:

- During installation: the installation fails.
- During an upgrade: the upgrade fails.
- During the deletion of an app: the deletion fails.

To view the dependencies for an app, from the **EDA Store** page, click the app's tile to open its page and click the **Required Applications** tab. As shown in the following example, requirements that are satisfied are checked.

*Figure 72: App requirements*

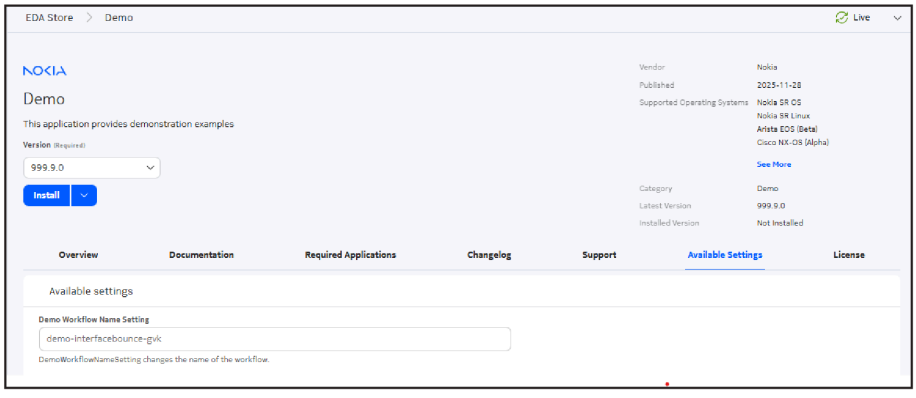



*sc0283*

## App settings

If an app has fields that you can set during installation, the **Available Settings** tab shows the fields that you can set.

*Figure 73: Available Settings tab*




*sc0284*

After app installation, the tab is renamed as **Applied Settings**.

### 14.3.1 Installing an application

#### About this task

After the EDA software is installed, EDA Store has access to the catalogs and registries. All apps available for installation are displayed in the EDA Store.
Use this procedure to install an app or execute a dry run of the app installation. The steps for the installation and dry-run installation are similar. An app installation dry run downloads the application image and applies the app manifest as a dry run transaction; this allows users to review the transaction result before proceeding with the app installation.

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.

#### Procedure

**Step 1.** From the **System Administration** navigation panel, select **Store**.

**Step 2.** Click the app that you want to install.

**Step 3.** In the app page that displays, you can install the app or a do a dry-run of the app installation:

- To install the app, click **Install**.
- To do a dry-run, click the drop-down list and select **Dry-run**

**Step 4.** If the app is available from more than one catalog, a form displays from which you can select the catalog. After making a selection, click **Install**.

**Expected outcome**

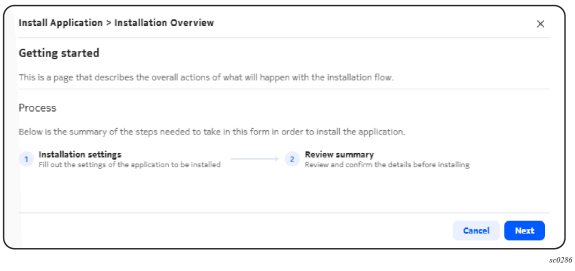
The flow of the installation varies depending on whether the app has requirements or associated settings:

- If the app has no outstanding requirements or associated settings, the system displays a **Summary** page with installation details and the procedure is complete.
- If the app has associated settings, the screens that display are shown in step 5. Follow the prompts to complete the installation.
- If the app has requirements, the screens that display are shown in step 6. Follow the prompts to complete the installation.

**Step 5.** If the app has settings associated with it, the system displays an **Installation Overview** page that describes the installation process for this app.

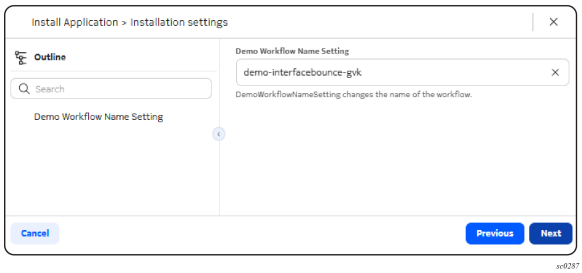
**a.** From the **Installation Overview** page, click **Next**.

*Figure 74: Installation Overview form*



**b.** In the **Installation settings** page that displays, set the fields for your app.

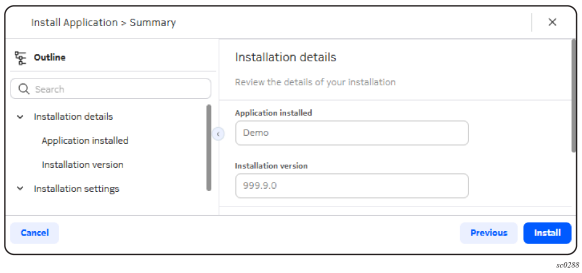
*Figure 75: Installation settings page*




Click **Next**.

**c.** From the **Summary** page, click **Install** to complete the installation.

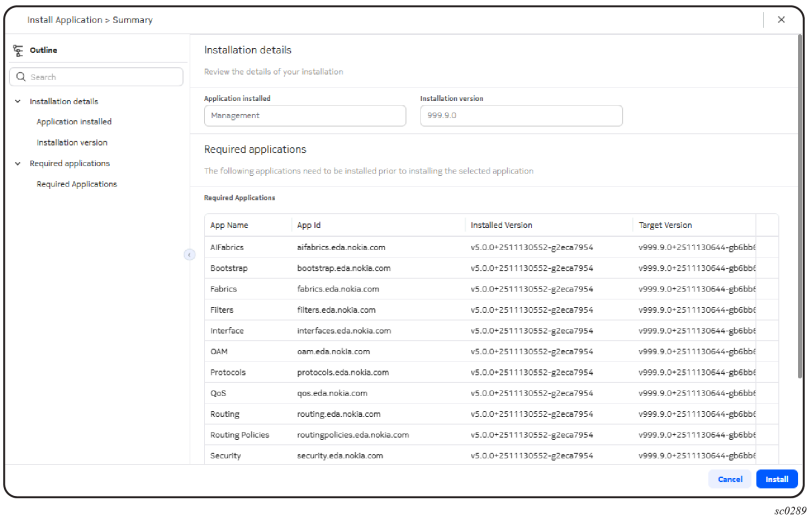
*Figure 76: Summary page for an app with settings*




The procedure is complete at this point.

**Step 6.** If an app has requirements associated with it, the **Summary** page displays required apps. Click **Install** or **Dry Run Install** to complete the procedure.

*Figure 77: Summary page for an app with requirements*



**Expected outcome**

The system installs the app and any app required identified in the **Summary** page.

**Related topics**

*Workflow Definition List page*

## 14.3.2 Editing app settings

### About this task

Use this procedure to update the settings of an app after installation. This procedure is applicable only if the application has settings associated with it, as described in App settings.

### Procedure

**Step 1.** From the **System Administration** navigation panel, select **Store**.

**Step 2.** Click the app for which you want to change settings, and click the **Applied Settings** tab.

**Step 3.** Click **Edit application settings**, then click **Next**.

**Step 4.** Update the needed settings.

Installation settings are unique to each app.

**Step 5.** Click **Install** to complete the procedure.

## 14.3.3 Uninstalling an app

### About this task

You can uninstall an app or do a dry run of uninstalling an app.

### Procedure

**Step 1.** From the **System Administration** navigation panel, select **Store**.

**Step 2.** Locate the app that you want to uninstall and click it.

**Step 3.** In the app page that displays, you can uninstall the app or do a dry-run of uninstalling the app:

- To uninstall the app, click **Uninstall**.
- To do a dry-run, click the drop-down list and select **Dry-run uninstall**

**Step 4.** Click **Uninstall** or **Dry run uninstall**.

**Note:** If you are trying to uninstall an app that is required for another installed app, the system displays an error.

## 14.3.4 Upgrading apps

### About this task

If an upgrade is available for an app, in EDA Store, the app tile displays the **Updates Available** button.

### Procedure

**Step 1.** From the **System Administration** navigation panel, select **Store**.

**Step 2.** Locate the app that you want to upgrade and click **Updates Available** on its tile.

**Step 3.** In the app page that opens, click the version to which you want to upgrade in the **Version** drop-down list.

The **Upgrade** button displays.

**Step 4.** You can update the app to the selected version or a do a dry-run installation of the selected version.

- To update the app, click **Install**.
- To do a dry-run of the upgrade, click **Dry-run** in the drop-down list.

**Step 5.** If the app is available from more than one catalog, a form displays from which you can select the catalog. After making a selection, click **Install**.

**Expected outcome**

The flow of the upgrade varies depending on whether the app has requirements or associated settings:

- If the app has no outstanding requirements or associated settings, the system displays a **Summary** page with installation details and the procedure is complete.
- If the app has associated settings, the screens that display are shown in step 6. Follow the prompts to complete the installation.
- If the app has requirements, the screens that display are shown in step 7. Follow the prompts to complete the installation.

**Step 6.** If the app has settings associated with it, the system displays an **Installation Overview** that describes the installation process for this app.

**a.** In the **Summary** page, click **Install**.

**b.** In the **Installation settings** page that displays, set the fields for your app.

**c.** From the **Summary** page, click **Install** to complete the installation.

**Step 7.** If an app has requirements associated with it, the **Summary** page displays required apps. Click **Install** to proceed with the installation.

## 14.4 Catalogs

A catalog is a Git repository that contains the manifests of applications. A manifest contains all the details of an app. EDA 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.

### 14.4.1 Creating a catalog credentials secret

#### About this task

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

### 14.4.2 Adding a catalog to the EDA Store

#### Prerequisites

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

#### Procedure

**Step 1.** From the **System Administration** navigation panel, select **Catalogs**.

**Step 2.** Click **Create**.

**Step 3.** Provide the following metadata for this resource:

- name
- labels
- annotations

**Step 4.** 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**
- **Refresh interval**: how often the controller checks the remote catalog for updates
- **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 5.** Click **Commit** to commit your change immediately or click **Add To Transaction** to add this item to transactions to commit later.

### 14.4.3 Managing a catalog

#### Procedure

**Step 1.** From the **System Administration** navigation panel, click **Catalogs**.

**Step 2.** Locate the catalog and click the action that you want to take from **Table row actions** menu.

You can take the following actions:

- view details about the catalog
- edit details about the catalog
- duplicate the catalog
- delete the catalog
  Before deleting a catalog, ensure that no apps are using it.

**Step 3.** Click **Commit** to commit your change immediately or click **Add To Transaction** to add this item to transactions to commit later.

## 14.5 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 EDA 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.

### 14.5.1 Creating the registry credentials secret

#### About this task

Before you can add a registry to the EDA 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
```

### 14.5.2 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

### 14.5.3 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

# 15 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.

## 15.1 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) for 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.<br>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.

### 15.1.1 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, 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*

#### 15.1.1.1 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: control 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/..`
- `/workflows/..`

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. `apiGroups` can include a * wildcard for the group version, for example, `core.eda.nokia.com/*`.
- **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.

To simplify user access to resource-related dashboards and queries, users are granted read permission to the following paths when they have the equivalent resource rules:

- `.namespace.resources.cr.{group}.{version}.{kind}.**`
- `.namespace.node.normal.{group}.{version}.{kind}.**`
- `.namespace.apps.cr.{group}.{version}.{kind}.**`

### 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. In the following example, the path includes all fields available directly under `admin`, but would not include child paths like `/core/admin/groups/{uuid}`:
  `/core/admin/*`
- **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 act 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 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.

## 15.1.1.2 Creating a cluster role

### About this task

A `ClusterRole` resource defines a set of permissions to access EDA resources.

### Procedure

**Step 1.** From the **System Administration** navigation panel, expand the **USER MANAGEMENT** group, then select **Cluster Roles**.

**Step 2.** If not already selected, click **Resources** from the **Cluster Roles** drop-down list.

**Step 3.** Click **Create**.

**Step 4.** Provide the following metadata for the `ClusterRole` resource:

- name

- namespace (if none is selected)
- labels
- annotations

**Step 5.** Provide an optional description for this `ClusterRole` resource.

**Step 6.** Create the list of resource rules.

In the **Resource Rules** section, click **+ Add**.

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

**b.** Click **None**, **read**, or **readWrite**in the **Permissions**drop-down list.

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 7.** 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 8.** 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
```

#### 15.1.1.3 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
```

#### 15.1.1.4 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. To create a role, go to the **System Administration** → **USER MANAGEMENT** menu and select **Roles**.

##### Procedure

**Step 1.** From the **System Administration** navigation panel, expand the **USER MANAGEMENT** group, then click **Roles**.

**Step 2.** If not already selected, click **Resources** from the **Roles** drop-down list.

**Step 3.** Click **Create**.

**Step 4.** Provide the following metadata for this resource:

- name
- namespace (if none is selected)
- labels
- annotations

**Step 5.** Provide an optional description for this cluster role.

**Step 6.** 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 **Add**.

**Step 7.** In the **Table Rules** section, click **+ Add**.

a. Provide the path to the database table to which this rule applies.
b. In the **Permissions** drop-down list, select **None** or **Read**.
c. Click **Save**.

**Step 8.** 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**.

**Step 9.** Click **Commit** to commit your change immediately or click **Add To Transaction** to add this item to transactions to commit later.

### 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: {}
```

## 15.1.2 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

### Viewing user groups

From the **System Administration** navigation panel, expand **USER MANAGEMENT** and click **User Management**. Select **User Groups** from the drop-down list.

**Note:** LDAP groups are displayed in the **User Groups** page only after they are imported from an LDAP server.

### 15.1.2.1 Creating a user group

#### Procedure

**Step 1.** From the **System Administration** navigation panel, expand the **USER MANAGEMENT** group, then click **User Management**.

**Step 2.** Click **User Groups** from the **User Management** drop-down list.

**Step 3.** Click **Create**.

**Step 4.** Provide a name for this user group.

**Step 5.** In the **Assigned Users** section, click **+ Add**.

Select the users that you want to assign to this user group, then click **Save**.

**Step 6.** From the **Assigned Roles** drop-down list, select a role to assign to the user group.

You can only select one role.

**Step 7.** Click **Save**.

### 15.1.2.2 Deleting user groups

#### Procedure

**Step 1.** From the **System Administration** navigation panel, expand the **USER MANAGEMENT** group, then click **User Management**.

**Step 2.** Click **User Groups** from the **User Management** drop-down list.

**Step 3.** Delete one or more user groups.

- To delete one user group, locate the user group that you want to delete and click **Delete** from the **Table row actions** menu.
- Alternatively, you can click one or more user groups. Then, click **Delete** from the **Table settings & actions** menu.

**Step 4.** Click **Save**.

### 15.1.3 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 with access 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 user

#### Users page

The **Users** page in the UI lists all local and remote EDA users and a provides a summary of user details. You can sort and filter for users using the typical mechanisms described in Working with data grids.

*Table 49: Default Users page display*

| Column | Description |
|---|---|
| Username | The login name for the user. |
| First Name | The first name of the user. |
| Last Name | The last name of the user. |
| Enabled | Indicates whether the user account is active or disabled. |

| Column | Description |
|---|---|
| Last Successful Login | The timestamp for the user's last successf login. |
| Federated User | The federation provider, if the user is a federation user. |
| Failed Logins Since Successful Login | The number of failed log in attempts after a user successfully logs in. This counter resets to 0 after the user successfully logs in again. |
| Last Failed Login | The timestamp for the user's last login failure. |
| Temporarily Disabled | Indicates if a user is temporarily disabled because of exceeding the allowed number of failed log in attempts. |

### 15.1.3.1 Creating a new local user

#### Procedure

**Step 1.** From the **System Administration** navigation panel, expand **USER MANAGEMENT** and select **User Management**.

**Step 2.** From the **User Management** drop-down list, click **Users**.

**Step 3.** Click **Create**.

**Step 4.** In the **User Information** section, enter the required information for the new user.

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

**Step 5.** Click **Set Password**.

In the form 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 6.** 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 7.** Click **Save**.

### 15.1.3.2 Managing user accounts

#### Prerequisites

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

#### Procedure

**Step 1.** From the **System Administration** navigation panel, expand **USER MANAGEMENT** and click **User Management**.

**Step 2.** Click **Users** from the **User Management** drop-down list.

**Step 3.** You can act on a single user or many users.

- To manage a single user, locate the user and click the action that you want to take from the **Table row actions** menu.
  - Click **Edit** to update details for a user such as user first name and last name, and assigned user groups. You can also enable or disable a user.
  - Click **Set Password** to set a new password.
- Alternatively, you can select more than one user, then click the **Table settings & actions button** icon. You can perform one of the following multi-row actions:
  - Delete the selected users.
  - Set passwords for the selected users.
  - Disable or enable the selected users.

**Step 4.** Click **Save**.

### 15.1.3.3 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**.

### 15.1.3.4 Restoring a user's default persistent settings

The **edactl aaa user settings clear** command restores a user's persistent settings in EDA to the default settings. This command is useful in scenarios where a user inadvertently writes an incompatible or errored change to their persistent settings, resulting in the browser going into a perpetual error loop.

The command does not impact dashboards and other user-generated content, but does impact settings for grid layouts, dark mode settings, and so forth.

## 15.1.4 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.

#### 15.1.4.1 Modifying the default password policy

**Procedure**

**Step 1.** From the **System Administration** navigation panel, expand **USER MANAGEMENT**, click **Password Policy**, then click **Edit**.

You can restore the default settings at this point or modify the password properties and lockout policy settings.

**Step 2.** Modify any of the following password properties:

- the minimum length of a password
- the minimum number of lowercase characters
- the minimum number of symbols or special characters
- the number of passwords to keep and validate against
- the minimum number of uppercase characters
- the minimum number of numerical characters
- whether the username can be used as a password
- the duration, in days, for a password to remain valid
- the hashing alogrithm: **ARGON2** (the default), **PBKDF2-SHA512**, **PBKDF2-SHA256**, or **PBKDF2**

**Step 3.** Modify any of the lockout policy settings:

- the maximum consecutive failed login attempts before account lockout
- duration, in seconds, to wait after reaching the maximum login failures before retry is allowed
- whether to lock the account permanently after maximum number of failed logins
- duration, in seconds, after which failed login attempts are reset

### 15.1.5 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.

Federated users are imported the first time a user logs in or when the user list is read via the EDA API/UI. Additionally, you can configure periodic sync of created and updates users.

In the EDA UI, federated users are identified in the **Federated Users** field in **User Management Users** page.

**Note:**
EDA only supports unsynchronized mode for Keycloak federation providers. This mode imports users and groups into EDA's Keycloak database, but does not write local changes back to the Lightweight Directory Access Protocol (LDAP) server.

The EDA API does not expose the full synchronization options from Keycloak to the federation provider. If full synchronization is required, it can be triggered via the Keycloak Administration Console.

EDA API server blocks all edits to federated users except for adding or removing the user to local groups. Local changes to federated groups are not supported; federated group membership must be configured on the LDAP server.

### Configuring remote directories

EDA supports:

- the configuration of up to five directories
- LDAP and Active Directory directories
- user synchronization from the directory
- group synchronization from the directory and user group membership mapping
- limiting imported users and groups using LDAP filters

When a remote directory is configured, system administrators can continue to create local users in EDA.

### Configuring TLS truststore for remote directories

When connecting a federation provider using LDAPS or STARTTLS, Keycloak must trust the server's TLS certificate authority. To add certificate authorities to the EDA Keycloak truststore, create a Kubernetes secret named `ldap-ca-secret` of type `Opaque` in the EDA base namespace with a base64 encoded PEM certificate in the `ca` field. For example:

```
apiVersion: v1
kind: Secret
type: Opaque
metadata:
  name: ldap-ca-secret
  namespace: eda-system #Enter the base namespace of your EDA installation
data:
  ca: <base64(certificate authority)> # Base64 encoded PEM certificate
```

EDA monitors this secret and if it changes, EDA updates the certificate authority information used by Keycloak. Modifying the authority information results in a restart of the Keycloak server.

## 15.1.5.1 Configuring a federation

### Procedure

**Step 1.** From the **System Administration** navigation panel, expand **USER MANAGEMENT** and select **User Management**.

**Step 2.** From the **User Management** drop-down list, click **Federations**.

**Step 3.** Click **Create**.

**Step 4.** Configure settings for this federation instance.

- a unique name
- the LDAP provider Vendor
- **Enabled**
- **Import Users**

**Note:** By default, this field is set to True; this field is ready-only.

**Step 5.** Configure LDAP server settings.

Set the following parameters:

- **Connection URL**
- **Use TLS**

**Note:** If this field is set to True, the certificate should be established on the LDAP server side. After configuring certificate from LDAP server, create the LDAP CA secret (`ldap-ca-secret`) on the platform where EDA is managed.

- **Bind Type**
- **User DN**
- **Username LDAP Attribute**
- **Timeout**
  Click **Test Connection** to test the connection to the LDAP server.
- **RDN LDAP Attribute**
- **ID Attribute**
- **User Object Classes**
- **User Search Filter**
- **Search Scope**
- **Pagination**
- **Periodic Sync**
- **Read Only**

**Step 6.** Enable and configure support for bind credentials.

Set the following parameters:

- **Bind Credential**
- **Bind DN**

Click **Test Authentication** to verify that the credentials are valid.

**Step 7.** Enable and configure group federation support.

If group support is disabled, groups are not synchronized with EDA. If group support is enabled, set the following parameters:

- **Object Classes**
- **Group LDAP DN**

- **Name LDAP Attribute**
- **Member Attribute**
- **Membership Attribute Type**
- **Membership User Attribute**
- **Filter**
- **Retrieval Strategy**
- **Member Of Attribute**

**Step 8.** Click **Save**.

#### 15.1.5.2 Deleting a federation

**Procedure**

**Step 1.** From the **System Administration** navigation panel, expand **USER MANAGEMENT** and click **User Management**.

**Step 2.** From the **User Management** drop-down list, click **Federations**.

**Step 3.** You can delete one federation or multiple federations at a time.

- Locate the federation that you want to delete and at click **Delete** from the **Table row actions** menu.
- Alternatively, you can select more than one federation, then, click the **Table settings & actions** menu on the upper right of the page and select **Delete**. Click **Save**.

**Step 4.** Click **Save**.

## 15.2 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 root CA 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

### 15.2.1 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.

#### 15.2.1.1 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
```

### 15.2.1.2 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 provide the API server 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
    Z05WQkFjVEIwTm9kVzh0YTNVeEVUQVBCZ05WQkFvVENFWnlZVzVyTkVSRQpNUmd3RmdZRFZRUUxF
    dzlYWldKRFpYSjBJRk4xY0hCdmNuUXhHREFXQmdOVkJBTVREMFp5WVc1ck5FUkVJRmRsCllpQkRR
    VEVqTUNFR0NTcUdTSWIzRFFFSkFSWVVjM1Z3Y0c5eWRFQm1jbUZ1YXpSa1pDNWpiMjB3SGhjTk1U
    TXcKTVRFeE1EUTFNVE01V2hjTk1UZ3dNVEV3TURRMU1UTTVXakJMTVFzd0NRWURWUVFHREFKS1VE
    RVBNQTBHQTFVRQpDQXdHWEZSdmEzbHZNUkV3RHdZRFZRUUtEQWhHY21GdWF6UkVSREVZTUJZR0Ex
    VUVBd3dQZDNkM0xtVjRZVzF3CmJHVXVZMjl0TUlHYU1BMEdDU3FHU0liM0RRRUJBUVVBQTRHSUFE
    Q0JoQUo5WThFaUhmeHhNL25PbjJTbkkxWHgKRHdPdEJEVDFKRjBReTliMVlKanV2YjdjaTEwZjVN
    Vm1UQllqMUZTVWZNOU1vejJDVVFZdW4yRFljV29IcFA4ZQpqSG1BUFVrNVd5cDJRN1ArMjh1bklI
    QkphVGZlQ09PekZSUFY2MEdTWWUzNmFScG04L3dVVm16eGFLOGtCOWVaCmhPN3F1TjdtSWQxL2pW
    cTNKODhDQXdFQUFUQU5CZ2txaGtpRzl3MEJBUVVGQUFPQmdRQU1meTQzeE15OHh3QTUKVjF2T2NS
    OEtyNWNaSXdtbFhCUU8xeFEzazlxSGtyNFlUY1JxTVQ5WjVKTm1rWHYxK2VSaGcwTi9WMW5NUTRZ
    RgpnWXcxbnlESnBnOTduZUV4VzQyeXVlMFlHSDYyV1hYUUhyOVNVREgrRlowVnQvRGZsdklVTWRj
    UUFEZjM4aU9zCjlQbG1kb3YrcE0vNCs5a1h5aDhSUEkzZXZ6OS9NQT09Ci0tLS0tRU5EIENFUlRJ
    RklDQVRFLS0tLS0K
  # base64 encoded private key
  tls.key: |
    RXhhbXBsZSBkYXRhIGZvciB0aGUgVExTIGNydCBmaWVsZA==
```

### 15.2.1.3 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 root CA (public and private keys) that EDA uses as an issuer for the node key-pairs. The user does this by creating a secret and a CA issuer that references the secret.

### 15.2.2 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

#### 15.2.2.1 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 to inject 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 and requests a certificate from Cert-Manager based on the `volumeAttributes` settings.

```
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
```

#### 15.2.2.2 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 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 without verifying the self-signed certificate. 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.

### Change in node issuer certificate triggers certificate rotation

When the node issuer certificate changes, 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 reaches the `CriticalFailedRotationThreshold`.

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.

## 15.3 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.

# 16 Platform security

This chapter covers topics related to securing the EDA infrastructure and managing internal passwords.

## 16.1 Changing internal passwords

EDA uses internal passwords to communicate between its internal services. These passwords are either hard-coded or are set before system installation.

After the system has been installed, administrators with cluster role privileges can update internal passwords for the following services using the applicable UI or scripts:

- Git passwords, Go Git server (Gogs) passwords
- Keycloak passwords and secrets
- PostgreSQL passwords

The following scripts are also available in the EDA toolbox pod:

- **reset-01-gogs-user-pass.sh**: resets the Gogs user password
- **reset-02-k8s-secret.sh**: resets the Kubernetes secret
- **reset-03-keycloak-admin-user.sh**: resets the Keycloak admin user password
- **reset-04-pgdb-password.sh**: resets the PostgreSQL database password

### 16.1.1 Updating the Git server password

#### Procedure

**Step 1.** Change the Git server password using the UI or the CLI script.

- Using the UI:
  - **a.** Log in to the Git UI. If you are using the git-server provided cluster, you can reach the UI using the following URLs:
    - `` `<eda-url>/core/httpproxy/v1/gogs/` ``
    - `` `<eda-url>/core/httpproxy/v1/gogs-replica/` ``
  - **b.** Click the user icon, then from the navigation bar on the right, go to **Your Settings** → **Password**.
  - **c.** Change your password.
  - **d.** Log out and then log back in.
- Using the CLI script:
  - **a.** Generate a token for the admin user.
    - **i.** Log in to the Git UI.
    - **ii.** Navigate to **Your Settings**.

iii. From the right sidebar, select **Applications** → **Generate New Token**. This token is required to access some admin-level REST endpoints.

**b.** Change the user password.
Open a shell to the EDA toolbox pod. The following example resets the user password for eda-git and eda-git-replica.

```
/eda/tools/reset-01-gogs-user-pass.sh \
-u eda \
-p oranges \
-g http://eda-git:3000 \
-t 79b6e0ada8dc74bf60751a0e56683d6377792070

/eda/tools//reset-01-gogs-user-pass.sh \
-u eda \
-p oranges \
-g http://eda-git-replica:3000 \
-t 70dd66f925678f35eb02d5073ce3b051b1bb640d
```

Where:
-u <username> is the username of the account

-p <password> is the new password for the user

-g <git server url> is the URL to reach the Gogs server

-t <access token> is the access token from an admin user

**Step 2.** Update the Gogs initialization secret.

**a.** Open a shell to the EDA toolbox pod.

**b.** Base64 encode the new password.

**Example**

```
echo -n "oranges" | base64
```

**c.** Change the gogs-admin-user secret.
The Gogs initialization secret is used on first boot of a new Gogs deployment.

Use the following command:

```
reset-02-k8s-secret.sh -n <namespace> -s gogs-admin-user -p <password>
```

Where:
`-n <namespace>` is the base namespace where EDA is deployed

`-p <password>` is the Base64 encoded new password for the user

**Example**

```
/eda/tools/reset-02-k8s-secret.sh -n eda-system -s gogs-admin-user -p b3Jhbmdlcw==
```

**Step 3.** Update the secret used by ConfigEngine.

**a.** Open a shell to the EDA toolbox pod.

**b.** Use the following command to change the Git secret:

```
reset-02-k8s-secret.sh -n <namespace> -s git-secret -p <password>
```

Where:
`-n <namespace>` is the base namespace where EDA is deployed

`-p <password>` is the new password for the user

**Example**

```
/eda/tools/reset-02-k8s-secret.sh -n eda-system -s git-secret -p oranges
```

**Step 4.** Restart the EDA cluster.

**a.** Open a shell to the EDA Toolbox.

**b.** Enter the follow commands to gracefully restart the cluster:

```
edactl platform stop
edactl platform start
```

### 16.1.2 Updating the Keycloak password

#### Procedure

**Step 1.** Change the Git server password using the UI or the CLI script.

- Using the UI:
  **a.** Go to the admin panel at `https://<domain:ip>/core/httpproxy/v1/keycloak`.
  **b.** Ensure that the Keycloak realm is selected from the upper left.
  **c.** Click **Users** from the left navigation bar
  **d.** Click the admin account.
  **e.** Click the **Credentials** tab, then click **Reset password** Follow the prompts to update the password.
- Using the CLI script:
  **a.** Open a shell to the EDA toolbox pod.
  **b.** Set a temporary Keycloak password.
  For example:

  ```
  /eda/tools/reset-03-keycloak-admin-user.sh -e https://eda-api -r admin -t
   temporary -a admin -p admin
  ```

  Where:
  -r <username> is the user for which to trigger a password reset

  -t <password> is the temporary password for the user

  -a <username> is the admin user to fetch an API token

  -p <password> is the admin user password to fetch an API token

  **c.** Log in to the Keycloak UI with the temporary password. Follow the prompts to update the password.

**Step 2.** Update the keycloak-admin-secret secret in Kubernetes

**a.** Open a shell to the EDA toolbox pod

**b.** The following example changes the keycloak-admin-secret secret:

```
/eda/tools/reset-02-k8s-secret.sh -n eda-system -s keycloak-admin-
secret -p oranges
```

Where:
-n <namespace> is the base namespace where EDA is deployed

-p <password> is the new password for the user

### 16.1.3 Update the PostgreSQL database using the script

#### About this task

Perform this procedure using the **reset-04-pgdb-password.sh** script from the toolbox pod.

#### Procedure

**Step 1.** Open a shell to the EDA toolbox pod.

**Step 2.** Update the database password.
Use the following command:

```
/eda/tools/reset-04-pgdb-password.sh -n <namespace> -p <password>
```

Where:

- `-n <namespace>` is the base namespace where EDA is deployed
- `-p <password>` is the new password for the user

**Example**

```
/eda/tools/reset-04-pgdb-password.sh -n eda-system -p oranges
```

**Step 3.** Update the Kubernetes secret password.

**Example**

```
/eda/tools/reset-02-k8s-secret.sh -n eda-system -s postgres-db-secret -p oranges
```

**Step 4.** Restart the Postgres and Keycloak deployments.

```
kubectl rollout restart deployment eda-postgres eda-keycloak
```

## 16.2 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 be unique for each installation. This practice is especially important for the Keycloak secrets that are used by the API server to configure and communicate with the Keycloak API server.

### 16.2.1 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.

### 16.2.2 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
```

# 17 Node RBAC

EDA supports the use of node RBAC to secure communication between EDA and nodes. System administrators can configure node security profile, node groups and node users using TACACS.

## 17.1 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 have 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 `operating System` 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 to 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 services (management services such as 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. The `NodeGroup` resource 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
```

## 17.1.1 Creating node groups

### Procedure

**Step 1.** From the **System Administration** navigation panel, expand **NODE MANAGEMENT**, then click **Node Groups**.

**Step 2.** If not already selected, click **Resources** from the **Node Users** drop-down list.

**Step 3.** Click **Create**.

**Step 4.** Provide the following metadata for this resource:

- name
- namespace (if none is selected)
- labels
- annotations

**Step 5.** Configure specifications for the node group.

- Provide a group name. If you do not provide a 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.
- Set the **Superuser** field to **True** to make members of this node user group superusers.

**Step 6.** 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 7.** If TACACS is used for authentication, in the **TACACS** section, select the privilege level.

**Step 8.** Click **Commit** to commit your change immediately or click **Add To Transaction** to add this item to transactions to commit later.

## 17.2 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

### 17.2.1 Creating node users

#### Procedure

**Step 1.** From the **System Administration** navigation panel, expand **NODE MANAGEMENT**, then click **Node Users**.

**Step 2.** If not already selected, click **Resources** from the **Node Users** drop-down list.

**Step 3.** Click **Create**.

**Step 4.** Provide the following metadata for the node user:

- name
- namespace (if none is selected)

- labels
- annotations

**Step 5.** Configure the specifications for this node user.

In the **Specification** section, provide a username and password for this user.

**Step 6.** 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 7.** In the **SSH Public Keys** field, click **Add item** to set the SSH public key to deploy for the user.

**Step 8.** Click **Commit** to commit your change immediately or click **Add To Transaction** to add this item to transactions to commit later.

### 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"
```

## 17.3 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:

- by 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 a `nodeSecurityProfile` CR 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 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"
```

# 18 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.

## 18.1 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.

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` resource.

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. You can set the following options:
  - `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.

**Related topics**

*Managing workflows with edactl*

## 18.1.1 Reimaging individual nodes

You can reimage individual nodes using the `Image` workflow or using the **edactl** tool.

### Example: Workflow resource for re-imaging specific nodes

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

- `nodes`: set to the name of the `TopoNodes` to be reimaged
- `nodeProfile`: set the Node Profile which contains the software image to use

```
apiVersion: os.eda.nokia.com/v1alpha1
kind: DeployImage
metadata:
  namespace: eda
  name: upgrade-leaf1
spec:
  nodes:
    - leaf-1
  nodeProfile: srlinux-25.7.1
```

## 18.1.2 Reimaging nodes using labels

You can reimage nodes using the `Image` workflow and applying labels to select TopoNodes or using the **edactl** tool.

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

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

- `nodeSelectors`: provide a list of label selectors to select `TopoNodes`.
- `nodeProfile`: set the node profile which contains the software image to use

```
apiVersion: os.eda.nokia.com/v1alpha1
kind: DeployImage
metadata:
  namespace: eda
  name: upgrade-rack1
spec:
  nodeSelectors:
    - eda.nokia.com/redundancy-group=rack1
```

```
  nodeProfile: srlinux-25.7.1
```

**Example: Using the edactl tool with a GVK workflow definition**

```
edactl workflow run operatingsystem-image-gvk workflow-fabric-upgrade-bylabel --bg -n eda
 type nodeselector nodeSelector[i] 0:maintenancenodes=fabric1 nodeProfile srlinux-24.10.3-
201 version 24.10.3 checks.skip true drains.skip true
```

### 18.1.3 Reimaging node tranches

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

- `tranches`: set to a list of `nodeSelector`, nesting the node selector type
- `canaries`: optionally used with tranches; specify as a pre-tranche to be reimaged before all others
- `nodeProfile`: set to the node `nodeProfile` resource that contains the software image to use

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. The tranche with index 0 is imaged next, following the same run-to-completion workflow.
3. The tranche with the next index is imaged next; this step repeats until all tranches have been upgraded.

**Example: Workflow resource for re-imaging tranches of nodes**

```
apiVersion: os.eda.nokia.com/v1alpha1
kind: DeployImage
metadata:
  namespace: eda
  name: upgrade-tranches
spec:
  tranches:
    - name: tranche1
      nodeSelectors:
        - eda.nokia.com/redundancy-group=rack1
    - name: tranche2
      nodeSelectors:
        - eda.nokia.com/redundancy-group=rack2
  nodeProfile: srlinux-25.7.1
```

### 18.1.4 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.

## 18.2 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 and run the `/tools/techsupport/techsupport.sh` script.

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.

**Related topics**

*Workflow Definition List page*

## 18.3 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 resource 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, 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.

By default, all repositories are backed up, but you can use the following options to exclude some repositories:

- **--exclude-identity-git-repo**: excludes the identity repository
- **--exclude-security-git-repo**: excludes the security repository
  Excluding the security repository prevents you from restoring an old security repository with expired certificates.

### 18.3.1 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.

### 18.3.2 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.

## 18.4 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. EDA takes this approach and is relevant for services like ConfigEngine. In general, any service should be able to restart and the system should converge back to a golden state. When any EDA pod fails, either Kubernetes or ConfigEngine should restart it.
- Localized redundancy and microservices: multiple instances of a common service with load balancing. This strategy limits localized outages, and 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, starts the active member, and then adds members back.

### Remote redundancy

Remote redundancy is accomplished by configuring a set of members within the `EngineConfig` resource in the `.spec.cluster.redundancy.members` 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:

- When there is failure to reach any member of the redundancy cluster
- When latency to a member is above a specified threshold
- Any core-generated alarms from any standby member
  These alarms are forwarded to the active member for the active to display, with the node set to the name of the member that raised it.

## 18.4.1 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.<br><br>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.<br><br>An operator must define the members of a geo-redundant cluster, where 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.

#### 18.4.1.1 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

  cluster:
    external:
      ipv4Address: 10.0.0.1
      ipv6Address: 2000::101
      domainName: cluster.eda.nokia.com
```

**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

  cluster:
```

```
    external:
      ipv4Address: 10.0.0.1
      ipv6Address: 2000::101
      domainName: cluster.eda.nokia.com

    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.

### 18.4.1.2 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
  cluster:
    external:
      ipv4Address: 10.0.0.1
      ipv6Address: 2000::101
      domainName: cluster.eda.nokia.com

    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`.

### 18.4.1.3 Migrating to the new Git server in the active cluster in standalone mode

#### Prerequisites

- Nokia recommends that you use the southbound interface for georedundancy configuration (port 51201).
- If the deployment has multiple active systems, that is, separate standalone systems, each system should point to a different set of Git servers.

#### Procedure

**Step 1.** Install two external Git servers.

Create the following repos in both Git servers:

- `/eda/customresources`
- `/eda/apps`
- `/eda/usersettings`
- `/eda/credentials`
- `/eda/identity`

**Step 2.** Put the TopoNodes at the standalone cluster in emulate mode.

**Step 3.** Back up the active cluster using the **edactl** command.

For instructions, see Creating backups.

**Step 4.** Stop the cluster.

**edactl stop**

**Step 5.** Update engine-config to point to the new set of Git servers.

**Step 6.** Restart the cluster.

**edactl start**

**Step 7.** Restore the backup files on the new Git servers.

**Step 8.** Put the TopoNodes at the standalone cluster in Normal mode.

### 18.4.1.4 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 the cluster member the local ConfigEngine. 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.
- `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. If no value is set, port 51201 is the default value used.

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

- The set of Git servers provided in the `.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 check ensures that standby clusters can take the load of the active cluster. This check occurs only initially while syncing a remote, as the values can change post run-time.
- The content of the `.spec.cluster` context must match. This includes members in `.spec.cluster.redundancy.members`, and information around external reachability of the cluster in the `.spec.cluster.external` context
- The content of `.spec.playground` and `.spec.simulate` must match.

### 18.4.1.5 Modifying the cluster name

#### About this task

Use this procedure to change the cluster name before the cluster operates geo-redundant mode.

#### Procedure

**Step 1.** Put the nodes in the cluster in emulate mode.

**Step 2.** Stop the cluster.

**Step 3.** Update the name of the resource in the `eda-kpt-base/engine-config/engineconfig.yaml` file.

**Step 4.** Apply the change to the cluster.

From the `eda-kpt-base` directory, enter the following command:

```
kpt live apply
```

### 18.4.1.6 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 platform
Name                Address     Port   ActivityState  BuildVersion                 Core
Version  AvgLatency(ms)  Reachable  Synchronized                         SyncedNpps  Synced
Nodes
kube-cp-a-cluster-1  self        51201  Active         v25.8.0-2508191403-g01589f68  v3.0.0
                  true       false
kube-cp-a-cluster-2  10.15.0.10  51201  Standby        v25.8.0-2508191403-g01589f68  v3.0.0
   3              true       false - WaitingForGitHashFromActive  10/10       N/A
root in on eda-toolbox-6584b57449-lb59m /eda/tools
```

#### 18.4.1.7 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, which 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.

## 18.5 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 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:

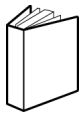
- `defaultRouterSelector`: specify a label
- `defaultRouters`: list the `DefaultRouters` resources

### 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