ETSI GS NFV-SOL 024 V5.3.1 (2025-09)

GROUP SPECIFICATION

Network Functions Virtualisation (NFV) Release 5; Protocols and Data Models; Specification of protocol and data model solutions for VNF Generic OAM functions and PaaS Services

Disclaimer

The present document has been produced and approved by the Network Functions Virtualisation (NFV) ETSI Industry Specification Group (ISG) and represents the views of those members who participated in this ISG.
It does not necessarily represent the views of the entire ETSI membership.

Reference
DGS/NFV-SOL024ed531

Keywords
API, data models, MANO, NFV, protocol

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI IPR online database.

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™**, **LTE™** and **5G™** logo are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM®** and the GSM logo are trademarks registered and owned by the GSM Association.

Istio®, Argo®, Prometheus® and Cilium® are registered trademarks of The Linux Foundation.

Foreword

This Group Specification (GS) has been produced by ETSI Industry Specification Group (ISG) Network Functions Virtualisation (NFV).

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The present document specifies protocol and data model solutions fulfilling the requirements specified in ETSI GS NFV-IFA 049 [1]. This present document analyses existing solutions against ETSI GS NFV-IFA 049 [1] requirements. Based on the analysis results, it develops the protocol and data model for the IF-F1 interfaces of VNF generic OAM functions and other PaaS Services, prominently using the CRD schema format where feasible, and other formats for some VNF generic OAM functions and other PaaS Services if CRD is not feasible. The present document specifies requirements for the protocol(s) on the IF-V interfaces toward the VNF. The resulting deliverable contains normative provisions.

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found in the ETSI docbox.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long-term validity.

The following referenced documents are necessary for the application of the present document.

[1] ETSI GS NFV-IFA 049: "Network Functions Virtualisation (NFV) Release 5; Architectural Framework; VNF generic OAM functions and other PaaS Services specification".

[2] Istio® v1.25.

[3] Fluent Bit v4.0.

[4] OpenSearch v3.0.0.

[5] Cilium® v1.17.

[6] Prometheus Alertmanager v0.28.l.

[7] OTEL Collector v0.126.0.

[8] VictoriaMetrics v1.116.0.

[9] VictoriaMetrics Operator v0.58.0.

[10] VictoriaMetrics Query API.

[11] Istio®: "Sail Operator v1.26.0".

[12] Fluent Bit: "Fluent Operator v3.4.0".

[13] OpenSearch: "OpenSearch Kubernetes® Operator v2.7.0".

[14] Prometheus Alertmanager: "Prometheus Operator v0.82.2".

[15] OpenSearch PPL API.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long-term validity.

The following referenced documents may be useful in implementing an ETSI deliverable or add to the reader's understanding, but are not required for conformance to the present document.

[i.1] ETSI GR NFV 003: "Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV".

[i.2] Kubernetes® API v1.32.

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in ETSI GR NFV 003 [i.1] apply.

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply.

ACL	Access Control List
CNCF	Cloud Native Computing Foundation
CRD	Custom Resource Definition
CRUD	Create Read Update and Delete
CSV	Comma-Separated Values
DNS	Domain Name System
DSL	Domain-Specific language
eBPF	extended Berkeley Packet Filter
gRPC	Google Remote Procedure Call
JSON	JavaScript Object Notation
JWT	JSON Web Token
kSQL	streaming SQL engine for Apache Kafka
LFN	Linux Foundation Networking
LogQL	Log Query Language
LTSV	Labelled Tab-Separated Values
mTLS	mutual Transport Layer Security
OAuth	Open Authorization
OIDC	OpenID Connect
OPA	Open Policy Agent
OTLP	OpenTelemetry Protocol
OTTL	OpenTelemetry Transformation Language
PPL	Piped Processing Language
PromQL	Prometheus Query Language
pub/sub	publish/subscribe
RBAC	Role Based Access Control
SDCIO	Schema Driven Configuration

SLO	Service Level Objective
SMI	Service Mesh Interface
SNI	Server Name Indication
SQL	Structured Query Language
SSL	Secure Sockets Layer
TLS	Transport Layer Security
UI	User Interface
YAML	YAML Ain't Markup Language

4 Overview

4.1 Introduction

This clause summarises the VNF generic OAM functions and PaaS services, along with their interfaces. It also gives an overview of the API structure, data models, and query APIs for the considered open-source solutions.

The present document covers the following VNF generic OAM functions and PaaS service:

- VNF generic OAM functions: Traffic Enforcer, Log Aggregator, Log Analyser, Metrics Analyser.
- PaaS Service: Notification Manager.

NOTE: Additional VNF generic OAM functions and PaaS services are out of the scope of the present document.

4.2 Summary of ETSI GS NFV-IFA 049

ETSI GS NFV-IFA 049 [1] specifies functional and interface requirements for VNF generic OAM functions and other PaaS Services: Table 4.2-1 lists the exposed interface and responsibilities of VNF generic OAM functions.

Table 4.2-1: Exposed interface and responsibilities of VNF generic OAM functions

VNF generic function	Exposed interface	Responsibilities
Traffic Enforcer	Traffic management interface (TrafficEnforcerInf)	To perform traffic isolation and rerouting of one or more VNFC instances.
Network Configuration Manager	Network configuration management interface (NetConfMaInf)	To configure the network connectivity for one or more VNF/VNFC instances with configuration data.
Upgrade VNF Manager	Upgrade VNF management interface (UpgVNFMaInf)	To coordinate the software modification or configuration of a VNF instance.
Log Aggregator	Log exposure interface (VNFLogAggregatorInf)	To aggregate, filter, expose logs collected from VNF instances or the NFVI and to store log records for further processing (e.g. for root-cause analysis).
Log Analyser	Log analysis exposure interface (VNFLogAnalyserInf)	To configure the processing of logs to be analysed and to expose log analysis results.
VNF Metrics Aggregator	Metrics exposure interface (VNFMetricAggregatorInf)	To aggregate, filter, expose metrics collected from VNF instances and to store time services metrics for further processing (e.g. abnormal behaviour detection).
VNF Metrics Analyser	Metrics analysis exposure interface (VNFMetricAnalyserInf)	To configure the processing of metrics to be analysed and to expose metrics analysis results.
Time Manager	Time management interface (TimeMaInf)	To configure the protocols used for time synchronization for VNF/VNFC instances and to expose logs related to time protocol operations on VNF/VNFC instances.
VNF Configuration Manager	VNF configuration management interface (VNFConfigMaInf)	To manage configuration of VNF/VNFC instances including but not limited to conveying virtualisation-dependent configuration items and configuration items at the application layer to VNF/VNFC instances.
VNF Testing Manager	VNF testing management interface (VNFTestingMaInf)	To manage multilayer testing of VNF instances including but not limited to setting testing plans, triggering the execution of tests and exposing testing results.

Table 4.2-2 lists the exposed interface and responsibilities of other PaaS Services.

Table 4.2-2: Exposed interface and responsibilities of other PaaS Services

Other PaaS Service	Exposed interface	Responsibilities
Configuration Server	Configuration data management interface (ConfigDataMaInf)	To store, update, delete configuration data in a logically centralized data repository and to process (e.g. validate) and expose configuration data set.
Notification Manager	Notifications management interface (NotificationMaInf)	To manage the subscriptions to notifications of events and to process (e.g. group) and expose notifications.
Policy Agent	Policy management interface (PolicyMaInf)	To manage and execute policies about VNF instances, VNF generic OAM functions and other PaaS Services.

4.3 Profiled protocols and data models for the selected open source solutions

4.3.1 Introduction

This clause provides an overview of the selected open source solutions that can be profiled based on the analysis in annex A of the present document. It provides an overview of the high-level API structures and the underlying data model concepts of the managed resource objects. Additionally, it provides an overview of the query APIs, which are profiled against query interface requirements of the Log and Metrics Analyser functions, as specified in ETSI GS NFV-IFA 049 [1].

4.3.2 API structure

This clause provides an overview of the selected open source solutions, including their high-level API structures and underlying data model concepts. Table 4.3.2-1 provides a high-level overview of the selected open source solutions, profiled against the generic OAM functions.

Table 4.3.2-1: High-level overview of the selected open source solutions profiled against the generic OAM functions

Generic OAM functions	Selected open source solutions	Overview
Traffic Enforcer function	Istio®	Istio® [2] is a service mesh that provides a way to manage microservices traffic, security, and monitoring.
	Cilium®	Cilium® [5] is a solution for networking, observability, and security, designed to improve visibility and control in containerized environments.
Log Aggregator function	Fluent Bit	Fluent Bit [3] is a log processor and forwarder for collecting, processing, and delivering log data.
	OpenTelemetry™ Collector	OpenTelemetry Collector [7] is a solution for providing a unified collection for telemetry data including logs.
Log Analyser function	OpenSearch	OpenSearch [4] is a search and analytics solution for real-time data analysis, visualization, and log management.
Metrics Analyser function	VictoriaMetrics	VictoriaMetrics [8] is a time-series database designed for collecting, storing, and querying metrics at scale.

Table 4.3.2-2 provides a high-level overview of the selected open source solutions, profiled against other PaaS Services.

Table 4.3.2-2: High-level overview of the selected open source solutions profiled against other PaaS Services

Other PaaS Services	Selected open source solutions	Overview
Notification Manager function	Prometheus Alertmanager	Prometheus Alertmanager [6] is a notification manager and forwarder for collecting, processing, and delivering notification data.

All the selected open source solutions use Kubernetes® native APIs and follow standard Kubernetes® conventions for resource management. They provide a set of declarative APIs that simplify the configuration and management of microservices behaviour.

These APIs follow standard RESTful principles and use standard terminology to describe resources:

- A **resource type** is the name used in the URL.
- All **resource types** have a representation in JSON (their object schema), called a **kind**.
- A collection represents a **list of instances** of a resource type.
- A single instance of a resource type is referred to as a **resource**, typically representing a configuration object.

Resources defined by these open source solutions are either **cluster-scoped** or **namespace-scoped** depending on their function and use case. The Kubernetes® API [i.2] supports read and write operations on resource objects of these open source solutions via Kubernetes® API endpoints. These custom resources extend the Kubernetes® API and are not included by default in standard Kubernetes® distributions; they can be installed and managed using their respective controllers or operators.

Each solution defines a set of custom resource objects to deliver its intended functionalities and configuration capabilities. For example, Istio® defines resources, such as **DestinationRule** and **AuthorizationPolicy** to manage service traffic, and enforce policies.

Standard HTTP methods (POST, PUT, PATCH, DELETE) are supported for operations on individual resources (or custom resources). However, the Kubernetes® API does not natively support submitting multiple resources as a part of a single transaction, whether ordered or unordered.

The mapping of the selected open source solutions' custom resource objects to the generic OAM functions and other PaaS Services, is provided in clauses 5 to 9 of the present document.

4.3.3 Data model concepts

Following the Kubernetes® [i.2] pattern for resource objects, the custom resource objects defined by the selected open source solutions are modelled using specific schemas. These resource definitions generally consist of the following four components:

- Custom Resource Kind: The **kind** field identifies the type of resource, such as **DestinationRule**, **AuthorizationPolicy** in the case of Istio®, or other solution-specific kind. It determines the category of configuration or behaviour the custom resource represents within the system.
- Custom Resource ObjectMeta: The **metadata** section contains information such as resource name, namespace, labels, and annotations. This metadata structure follows a schema common to all Kubernetes® resources. Certain metadata fields, such as annotations, may be modified by users or solution specific controllers, while fields like name and namespace are typically immutable after resource creation.
- Custom Resource Spec: The **spec** section is defined by the user and describes the desired state of the resource. This field is defined during the creation or modification of the resource to specify its intended behaviour and configuration.
- Custom Resource Status: The **status** section summarizes the current state of the system concerning the managed custom resource object.

These components form the core of the declarative model used by all selected open source solutions to manage resource lifecycle and configuration in a Kubernetes-native manner.

4.3.4 Query APIs for the Log and Metrics Analyser functions

4.3.4.1 Introduction

This clause provides an overview of the query APIs, which are profiled against query interface requirements of the Log and Metrics Analyser functions, as specified in ETSI GS NFV-IFA 049 [1].

4.3.4.2 OpenSearch PPL API

The OpenSearch PPL API [4] provides a RESTful, operation-focused interface for querying structured log data, optimized for log analytics and search operations. Unlike Kubernetes-based OpenSearch resource management, which follows a declarative model to define resources, the PPL API enables users to execute dynamic, ad-hoc queries against indexed log data using a SQL-like syntax. It does not represent a Kubernetes® resource or alter OpenSearch configurations; rather, it offers a flexible querying mechanism designed specifically for retrieving and analysing log data.

All PPL queries are executed via the "/_plugins/_ppl" endpoint using an HTTP POST request with a JSON-based query body. The request body shall contain a valid PPL query, and the response returns a structured JSON object with the query results.

4.3.4.3 VictoriaMetrics Query API

VictoriaMetrics provides a Query API [10] designed to retrieve and process time-series metrics data for monitoring, alerting, and visualization. This API is Prometheus-compatible and leverages MetricsQL (an extended PromQL syntax) to execute operations such like filtering, aggregation, and rate calculations on metrics stored in VictoriaMetrics.

The API operates in read-only mode; it does not modify stored data or system configurations. Instead, it enables dynamic querying of metric data through endpoints "/api/v1/query" for retrieving metrics at a specific timestamp, and "/api/v1/query_range" for retrieving metrics over defined time ranges.

Queries can be executed via HTTP GET or POST requests. GET requests pass parameters in the URL, while POST requests accept URL-encoded form data (in key=value pairs) in the request body. The API returns results in structured JSON format, suitable for consumption by tools like Grafana® or for automated processing in external systems.

5 VNF generic OAM functions object models mapping to profiled solution objects

5.1 Traffic Enforcer object mapped to Istio®

The selected Istio® [11] custom resource objects are identified to map to the Traffic Enforcer object of the NFV object model, see clause 6.3.1 in ETSI GS NFV-IFA 049 [1]. Table 5.1-1 lists the custom resource objects which are mapped to the input and output parameters of the Traffic Management Interface in ETSI GS NFV-IFA 049 [1].

Table 5.1-1: Istio® custom resource object mapped to Traffic Enforcer object

Istio® custom resource object kind	Istio® custom resource URI	Istio® custom resource object description
Gateway	/apis/networking.istio.io/v1/namespaces/{namespace}/gateways	Namespaced-resource that defines a load balancer operating at the edge of the mesh receiving incoming or outgoing HTTP/TCP connections. The specification describes a set of ports that should be exposed, the type of protocol to use, Server Name Indication (SNI) configuration for the load balancer, etc.
VirtualService	/apis/networking.istio.io/v1/namespaces/{namespace}/virtualservices	Namespaced-resource that defines how requests are routed to different services within the mesh. It allows to configure traffic routing rules like route weighting and traffic shifting, fault injection, HTTP routes, TCP routes, gateways, and advanced routing based on request headers or other criteria. For example, it can direct traffic to specific versions of a service for canary deployments or A/B testing.
DestinationRule	/apis/networking.istio.io/v1/namespaces/{namespace}/destinationrules	Namespaced-resource that specifies policies that apply to traffic after it has been routed to a specific service. It allows fine-grained control over aspects such as load balancing, timeouts, retries, and connection pool settings for a particular service.

Istio® custom resource object kind	Istio® custom resource URI	Istio® custom resource object description
AuthorizationPolicy	/apis/security.istio.io/v1/namespaces/{namespace}/authorizationpolicies	Namespaced-resource that governs access control within the service mesh by specifying who can access a service and under what conditions. It uses attributes like source IP, request headers, and JWT tokens to allow or deny traffic, enabling detailed security policies for services in the mesh.
NOTE: Clause 6.2.1 of ETSI GS NFV-IFA 049 [1] specifies requirements for traffic isolating, blocking and routing, which can be implemented using the Istio® custom resources referenced. Specifically, traffic routing to specific VNFC instances can be managed using a DestinationRule, following the creation of a VirtualService and a Gateway. Similarly, an AuthorizationPolicy can be used to block or isolate specific VNFC instances.		

See the following for an end-to-end explanation of traffic routing with Istio®:

- In Istio®, when a user (or operator) applies CRDs such as Gateway, VirtualService, DestinationRule, and AuthorizationPolicy, Istiod reads these CRDs, generates proxy-specific configurations, and pushes them to the appropriate Envoy proxies in the data plane, such as the Ingress Gateway and sidecar proxies running alongside service pods.

When a user sends an external request (e.g. GET), it first reaches the Ingress Gateway, which exposes external traffic to the service mesh, listens on ports like 443, provides TLS termination, and forwards the request based on rules defined in a VirtualService - such as those matching hostnames or paths. A DestinationRule then configures how the traffic is sent to specific service versions by defining subsets (e.g. v1) using labels and applying load balancing policies (e.g. ROUND_ROBIN). These subsets typically map to pods running particular versions of a service. Before the request reaches the application, the sidecar proxy inside the pod enforces security using an AuthorizationPolicy, checking conditions like a valid JWT token, request method, or request path.

All of these configurations are centrally managed by Istiod, which continuously reads the CRDs, compiles them into proxy-specific rules, and distributes them to sidecar proxies and gateways.

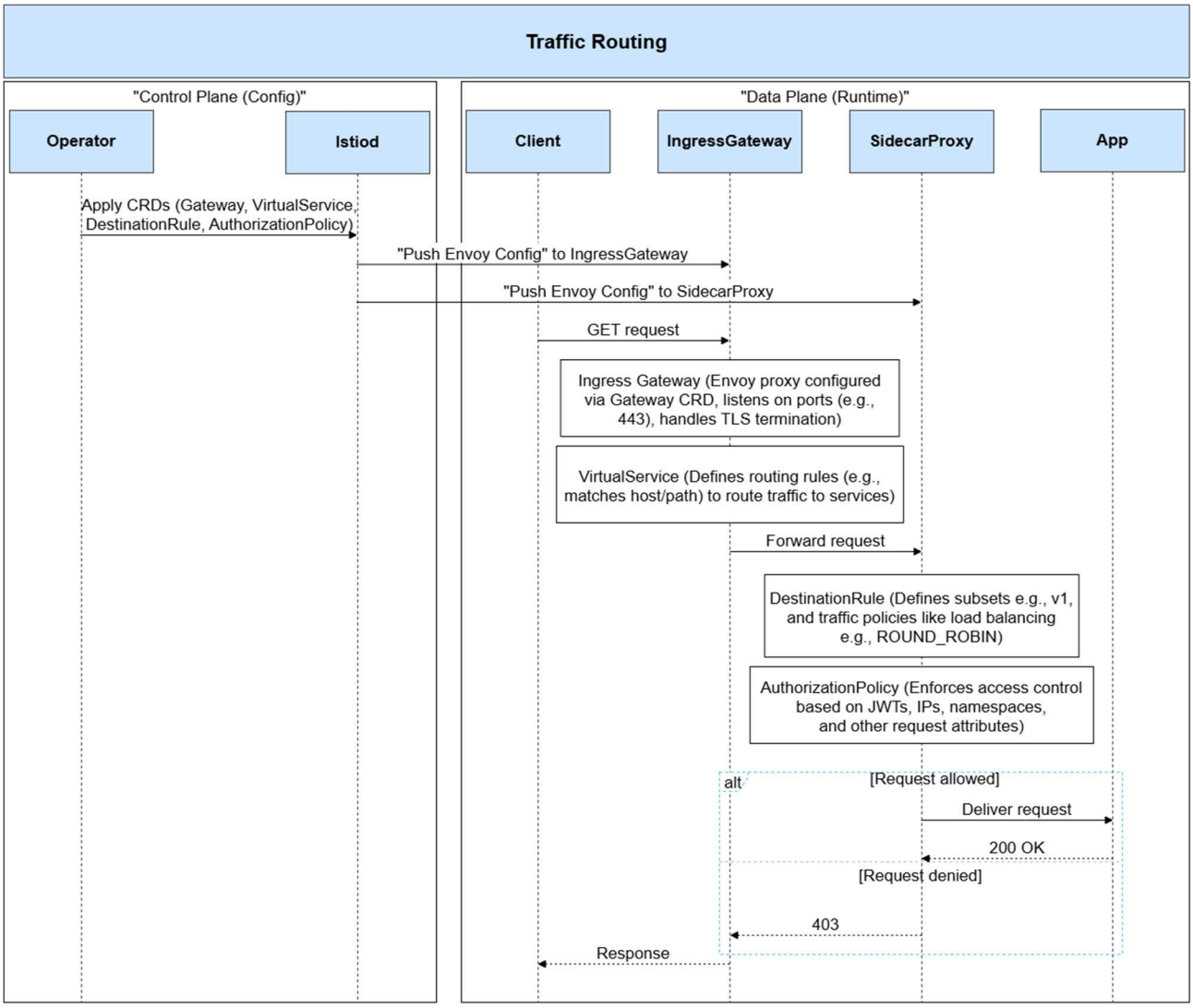

Figure 5.1-1: Traffic routing with Istio® using Gateway, VirtualService, DestinationRule, and AuthorizationPolicy

5.1a Traffic Enforcer object mapped to Cilium®

The selected Cilium® [5] custom resource objects are identified to map to the Traffic Enforcer object of the NFV object model, see clauses 5.2, 6.2.1 and 6.3.1 in ETSI GS NFV-IFA 049 [1]. Table 5.1a-1 lists the Cilium® custom resource objects which are mapped to the input and output parameters of the Traffic Management Interface in ETSI GS NFV-IFA 049 [1].

Table 5.1a-1: Cilium® custom resource object mapped to Traffic Enforcer object

Cilium® custom resource object kind	Cilium® custom resource URI	Cilium® custom resource object description
CiliumNetworkPolicy	/apis/cilium.io/v2/namespaces/{namespace}/ciliumnetworkpolicies	Namespaced resource that defines network policies for controlling traffic at L3/L4 and L7 layers (see note).
CiliumClusterNetworkPolicy	/apis/cilium.io/v2/ciliumclusterwidenetworkpolicies	Cluster-scoped resource that defines global network policies that apply across all namespaces.
CiliumEnvoyConfig	/apis/cilium.io/v2/namespaces/{namespace}/ciliumenvoyconfigs	Namespaced resource that enables advanced L7 traffic management capabilities by applying custom Envoy proxy configurations.
NOTE: To support partial traffic isolation, this custom resource can be applied to block all new incoming connections to a Kubernetes® pod while keeping established TCP connections to other pods of a workload application.		

5.2 Log Aggregator object mapped to Fluent Bit

The selected Fluent Bit [12] custom resource objects are identified to map to the Log Aggregator object of the NFV object model, see clause 6.3.4 in ETSI GS NFV-IFA 049 [1].

Table 5.2-1 lists the Fluent Bit custom resource objects which are mapped to the input and output parameters of the Log Aggregator Exposure Interface as described in ETSI GS NFV-IFA 049 [1].

Table 5.2-1: Fluent Bit custom resource object mapped to Log Aggregator object

Fluent Bit custom resource object kind	Fluent Bit custom resource URI	Fluent Bit custom resource object description
ClusterInput	/apis/fluentbit.fluent.io/v1alpha2/clusterinputs	Cluster-scoped resource that defines the configuration for collecting log data from various sources within the Kubernetes® cluster. It specifies the log input sources, such as file paths or system logs, and determines how logs are gathered and processed before further transformation or routing.
ClusterParser	/apis/fluentbit.fluent.io/v1alpha2/clusterparsers	Cluster-scoped resource that defines how to extract structured data from raw log lines. It specifies the format of the log entries (e.g. JSON, logfmt, custom regex) and which fields to extract. ClusterParser resources is typically used by input plugins to parse unstructured logs before they are passed through filters such as ClusterFilter.
ClusterFilter	/apis/fluentbit.fluent.io/v1alpha2/clusterfilters	Cluster-scoped resource that configures filters to process and transform log data as it flows through the Fluent Bit pipeline. ClusterFilter resources can enrich logs with metadata, remove sensitive information, add or modify log fields, and filter logs based on specific criteria.
ClusterOutput	/apis/fluentbit.fluent.io/v1alpha2/clusteroutputs	Cluster-scoped resource that configures the destination for processed log data. It defines the output plugin type, target endpoint, and any necessary authentication or authorization details. ClusterOutput resources are used to send logs to various destinations for analysis, storage, or alerting.
NOTE: Clause 6.2.4 of ETSI GS NFV-IFA 049 [1] specifies requirements for log filtering and querying. These requirements can be fulfilled using Fluent Bit's custom resources. To manage log filtering, the Filter custom resource can be used, following the creation of ClusterInput and Parser custom resources to ingest and structure the logs. To expose logs to various destinations for analysis, storage, or visualization, the Output custom resource can be used.		

See the following for an end-to-end explanation of log aggregation with Fluent Bit:

- Fluent Bit collects logs generated by containers, files, or systemd using a DaemonSet that runs on each Kubernetes® node. As a first step, the ClusterInput resource defines how and from where logs are collected. Raw log entries are then passed to a ClusterParser, which transforms unstructured logs into structured data using formats like JSON, regex, or multiline. The parsed logs are then passed through a ClusterFilter resource, which can enrich, modify, or filter log entries, for example, by adding Kubernetes® metadata or removing sensitive fields. Finally, the processed logs are routed to their destination as specified by a ClusterOutput, such as Elasticsearch, S3, or any other custom endpoint.

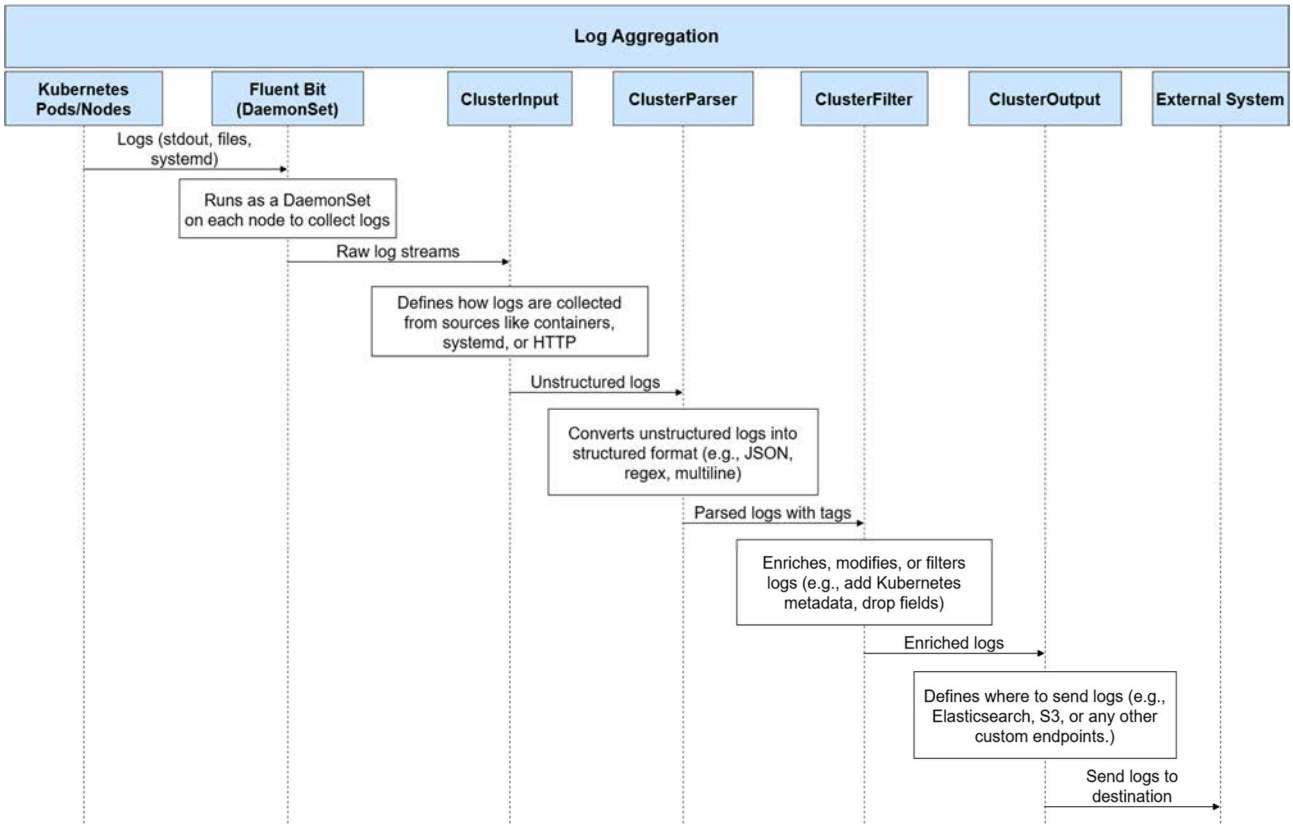

Figure 5.2-1: Log aggregation with Fluent Bit using ClusterInput, ClusterParser, ClusterFilter, and ClusterOutput

5.3 Log Analyser object mapped to OpenSearch

The selected OpenSearch [13] custom resource objects are identified to map to the Log Analyser object of the NFV object model, see clause 6.3.5 in ETSI GS NFV-IFA 049 [1].

Table 5.3-1 lists the OpenSearch custom resource objects which are mapped to the input and output parameters of the Log Analysis Exposure Interface as described in ETSI GS NFV-IFA 049 [1].

Table 5.3-1: OpenSearch custom resource object mapped to Log Analyser object

OpenSearch custom resource object kind	OpenSearch custom resource URI	OpenSearch custom resource object description
OpensearchComponentTemplate	/apis/opensearch.opster.io/v1/namespaces/{namespace}/opensearchcomponenttemplates	Namespaced resource that defines a blueprint for how OpenSearch indexes should be structured. An OpenSearch index is essentially a collection of documents, like a database table, where each document represents a unit of information. This blueprint allows defining reusable configurations, such as how many shards and replicas each index should have, and how data within your indexes should be organized and searched. Instead of setting or configuring every single index, this template can be created once and apply it to many indexes, to save time and effort. For example, in context of the logs analysis, once logs are ingested, they are stored in indexes for fast retrieval and filtering.
OpensearchIndexTemplate	/apis/opensearch.opster.io/v1/namespaces/{namespace}/opensearchindextemplates	Namespaced resource that serves as a blueprint for a group of indexes that follow a specific naming pattern (like logs-2020-01-*). It streamlines index management by automatically applying predefined configurations from component templates to all matching indexes. By defining reusable rules, the OpenSearchIndexTemplate ensures consistency, reduces manual effort, and minimizes configuration errors. This approach simplifies index lifecycle management, improves efficiency, and helps maintain a well-structured OpenSearch environment.
OpenSearchISMPolicy	/apis/opensearch.opster.io/v1/namespaces/{namespace}/opensearchismpolicies	Namespaced resource that automates the lifecycle management of indexes, particularly for time-series data where older data becomes less relevant over time. It enables automated actions such as reducing replica counts, transitioning indexes to a read-only state, or deleting outdated indexes based on factors like index age, size, or document count. For example, a policy can move an index to read-only after 30 days and delete it after 90 days. Leveraging this CRD streamlines data management, optimizes resource utilization, and ensures that data remains accessible and relevant based on specific criteria.
OpenSearchCluster	/apis/opensearch.opster.io/v1/namespaces/{namespace}/opensearchclusters	Namespaced resource that simplifies the deployment and management of a complete OpenSearch environment by automating the creation of OpenSearch cluster along with OpenSearch Dashboards. It sets up the necessary infrastructure for log visualization, analysis, anomaly detection, and alerting. OpenSearch Dashboards also provide an efficient interface for querying and exploring data with various query languages, supporting efficient log management, analytics, and real-time monitoring with customizable notifications and alerts.
NOTE: Clause 6.2.5 of ETSI GS NFV-IFA 049 [1] specifies requirements for log processing and exposing log analysis results. These requirements can be fulfilled using the referenced OpenSearch custom resources. Log processing and analysis visualisation can be managed through the OpenSearch dashboards, which become accessible after deploying the OpenSearchCluster custom resource. This requires the prior creating of supporting resources, including OpenSearch Data Prepper (solution for log ingestion) and other custom resources OpensearchComponentTemplate, OpensearchIndexTemplate, and OpenSearchISMPolicy.		

See the following for an end-to-end explanation of log analysis and querying with OpenSearch:

- When a user (or operator) applies the OpenSearchCluster CRD, it deploys and manages the OpenSearch nodes. Applying additional OpenSearch CRDs provides the supporting configurations: the OpensearchComponentTemplate CRD defines reusable index settings and mappings; the OpensearchIndexTemplate CRD specifies index patterns and links them to component templates; and the OpenSearchISMPolicy CRD configures index lifecycle rules such as rollover and deletion. Once the cluster is running and templates are in place, log agents or applications send data to the OpenSearch cluster, which automatically creates indices based on the defined templates and policies. Users can perform log analysis by running SQL-like queries via the PPL API (_plugins/_ppl), which translates the query into OpenSearch DSL, executes it on the relevant log indices, and returns the results in a structured JSON format.

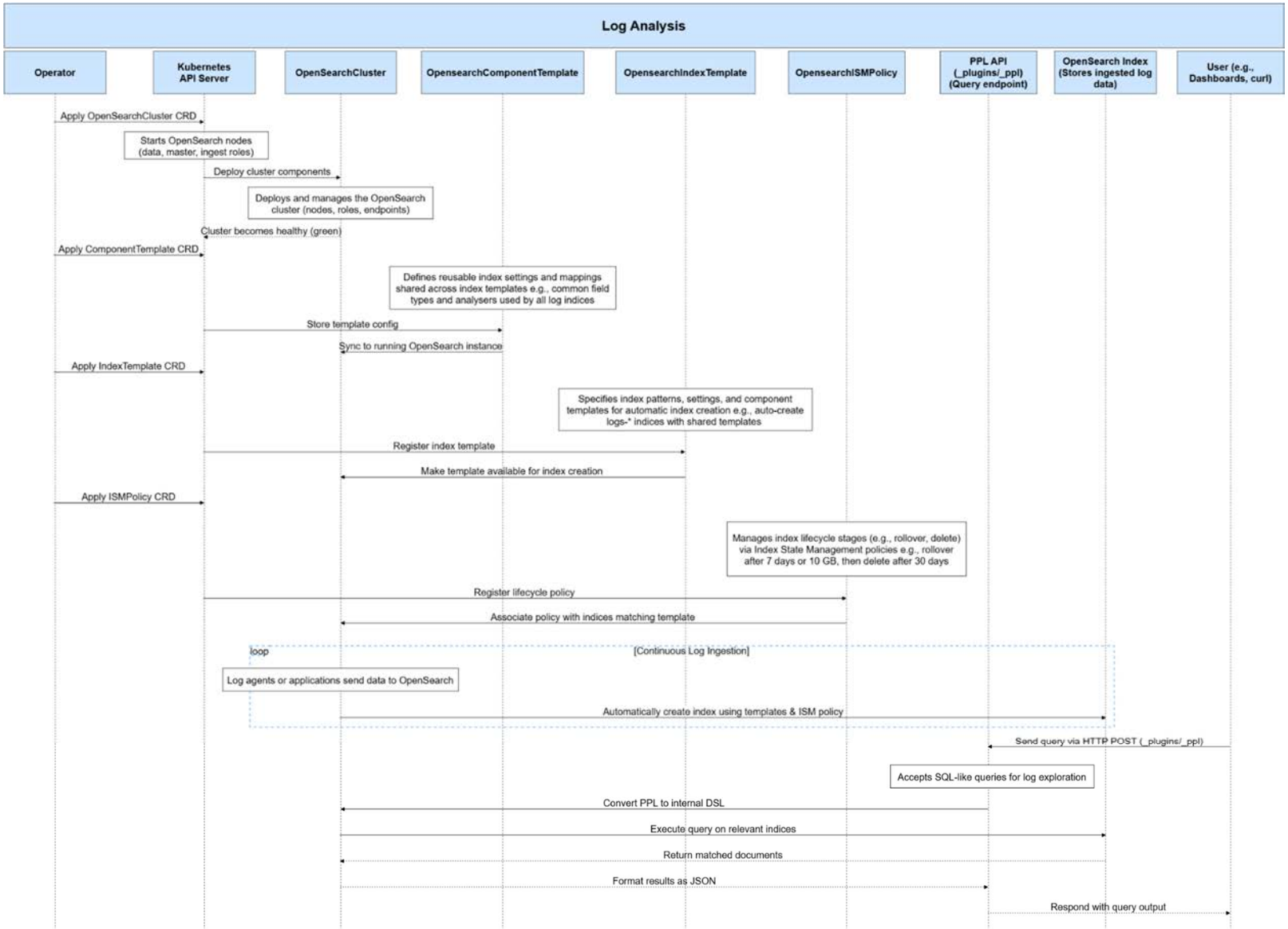

Figure 5.3-1: Log analysis with OpenSearch using OpenSearchCluster, OpensearchComponentTemplate, OpensearchIndexTemplate, and OpenSearchISMPolicy

5.4 Notification Manager object mapped to Prometheus Alertmanager

The selected Prometheus Alertmanager [14] custom resource objects are identified to map to the notification manager object of the NFV object model, see clause 6.3.12 in ETSI GS NFV-IFA 049 [1].

Table 5.4-1 lists the OpenSearch custom resource objects which are mapped to the input and output parameters of the Log Analysis Exposure Interface as described in ETSI GS NFV-IFA 049 [1].

Table 5.4-1: Prometheus Alertmanager custom resource object mapped to Notification manager object

Prometheus Alertmanager custom resource object kind	Prometheus Alertmanager custom resource URI	Prometheus Alertmanager custom resource object description
PrometheusRule	/apis/monitoring.coreos.com/v1/namespaces/{namespace}/prometheusrules	Namespaced resource that defines alerting and recording rules for Prometheus. It evaluates scraped data and triggers alerts when specified conditions are met, which are then processed by Alertmanager for routing.
AlertmanagerConfig	/apis/monitoring.coreos.com/v1alpha1/namespaces/{namespace}/alertmanagerconfigs	Namespaced resource that defines custom alert routing rules and notification configurations for Alertmanager. It specifies how alerts should be grouped, filtered, and sent to different receivers (e.g. email, webhook). This resource allows multiple teams or users to configure independent notification settings within a shared Alertmanager instance.
Alertmanager	/apis/monitoring.coreos.com/v1/namespaces/{namespace}/alertmanagers	Namespaced resource that represents an instance of Prometheus Alertmanager responsible for managing alerts, deduplicating them, grouping them, and routing them to defined notification channels. This resource defines global configurations for alert processing, notification handling, and inhibition rules.

See the following for an end-to-end explanation of notification management using Prometheus Alertmanager:

- User (or operator) subscribes to notifications by applying the AlertmanagerConfig CRD. This CRD serves as the control plane for notification routing, defining critical parameters such as receiver endpoints (e.g. webhook URLs), alert routing rules based on severity or labels, and inhibition logic to prevent duplicate alerts. Meanwhile, PrometheusRule CRD establishes the alert conditions that monitor system metrics, effectively determining when specific events should trigger alerts. When these predefined conditions are met, Prometheus® generates alerts and forwards them to Alertmanager for processing. Alertmanager evaluates each alert against the configured routing and inhibition rules to determine whether it should be sent or suppressed. If allowed, the alert is forwarded to the defined receiver (e.g. webhook, email). Otherwise, it is dropped to avoid unnecessary noise or duplication.

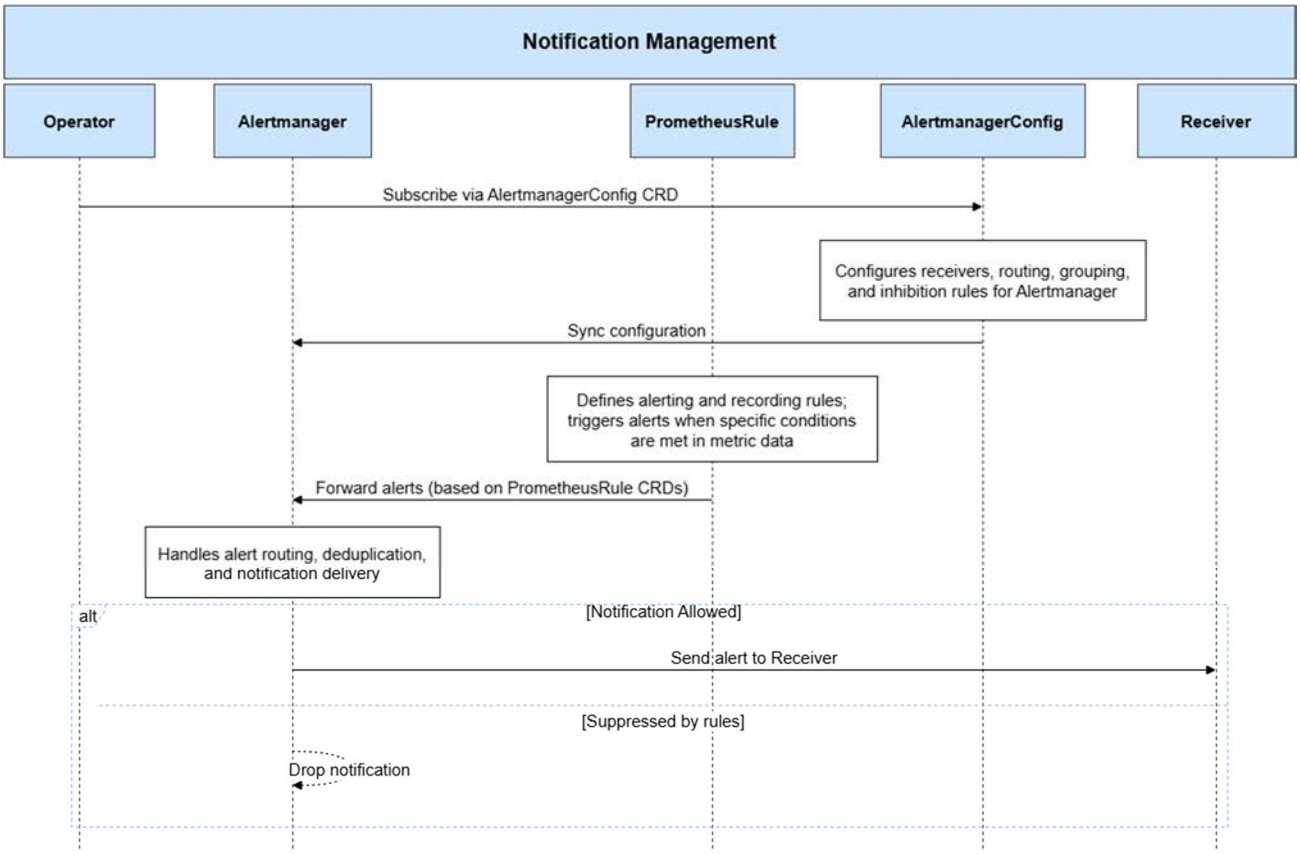

Figure 5.4-1: Notification management with Prometheus Alertmanager using Alertmanager, AlertmanagerConfig, and PrometheusRule

5.5 Metrics Analyser object mapped to VictoriaMetrics

Selected VictoriaMetrics [9] custom resource objects are identified to map to the Metrics Analyser object of the NFV object model, see clause 6.3.7 in ETSI GS NFV-IFA 049 [1]. Table 5.5-1 lists the VictoriaMetrics custom resource objects which are mapped to the input and output parameters of the Metrics Analyser Interface in ETSI GS NFV-IFA 049 [1].

Table 5.5-1: VictoriaMetrics custom resource object mapped to Metric Analyser object

VictoriaMetrics custom resource object kind	VictoriaMetrics custom resource URI	VictoriaMetrics custom resource object description
VMAgent	/apis/operator.victoriametrics.com/v1beta1/namespaces/{namespace}/vmagents	Namespaced resource that defines how metrics are scraped from endpoints and forwarded to VictoriaMetrics for storage and analysis.
VMRule	/apis/operator.victoriametrics.com/v1beta1/namespaces/{namespace}/vmrules	Namespaced resource that defines alerting and recording rules to evaluate. It defines MetricsQL-based expressions to analyze time-series data stored in VictoriaMetrics, enabling automated issue detection or metric generation. VMRule allows the configuration of alerts for monitoring and observability workflows.
VMAlert	/apis/operator.victoriametrics.com/v1beta1/namespaces/{namespace}/vmalerts	Namespaced resource that evaluates VMRule definitions against VictoriaMetrics data. It processes configured rules at fixed intervals, triggers alerts when conditions are met, and forwards them to VMAlertmanager for notification routing.
VMAlertmanager	/apis/operator.victoriametrics.com/v1beta1/namespaces/{namespace}/vmalertmanagers	Namespaced resource that defines the deployment and configuration of alert routing, inhibition, and notification delivery based on received alerts.
NOTE: Clause 6.2.7 of ETSI GS NFV-IFA 049 [1] defines requirements for metrics processing and exposure, which can be fulfilled using VictoriaMetrics resources. The VMRule defines alerting and recording rules for metrics evaluation, while VMAlert executes these rules against data stored in VMCluster and forwards any resulting alerts to VMAlertmanager for notification routing and delivery. For external access, VictoriaMetrics also provides a Prometheus-compatible Query API to query time-series metrics data.		

See the following for an end-to-end explanation of metrics analysis and querying with VictoriaMetrics:

- An operator first defines evaluation rules using the VMRule resource, specifying expressions and thresholds for detecting conditions like high CPU usage or latency spikes. Alerting logic such as duration, severity, and labels is configured through VMAlert, while notification routing and inhibition policies are managed by VMAlertmanager, which loads its config from VMAlertmanagerConfig. The VMAgent continuously scrapes metrics and evaluates them against the defined rules. When an alert condition is met, VMAlert triggers an alert and forwards it to VMAlertmanager. Based on the routing policy, the alert is either sent to the configured receiver or suppressed. Metrics consumers can also query analytics results via the standard "/api/v1/query" endpoint.

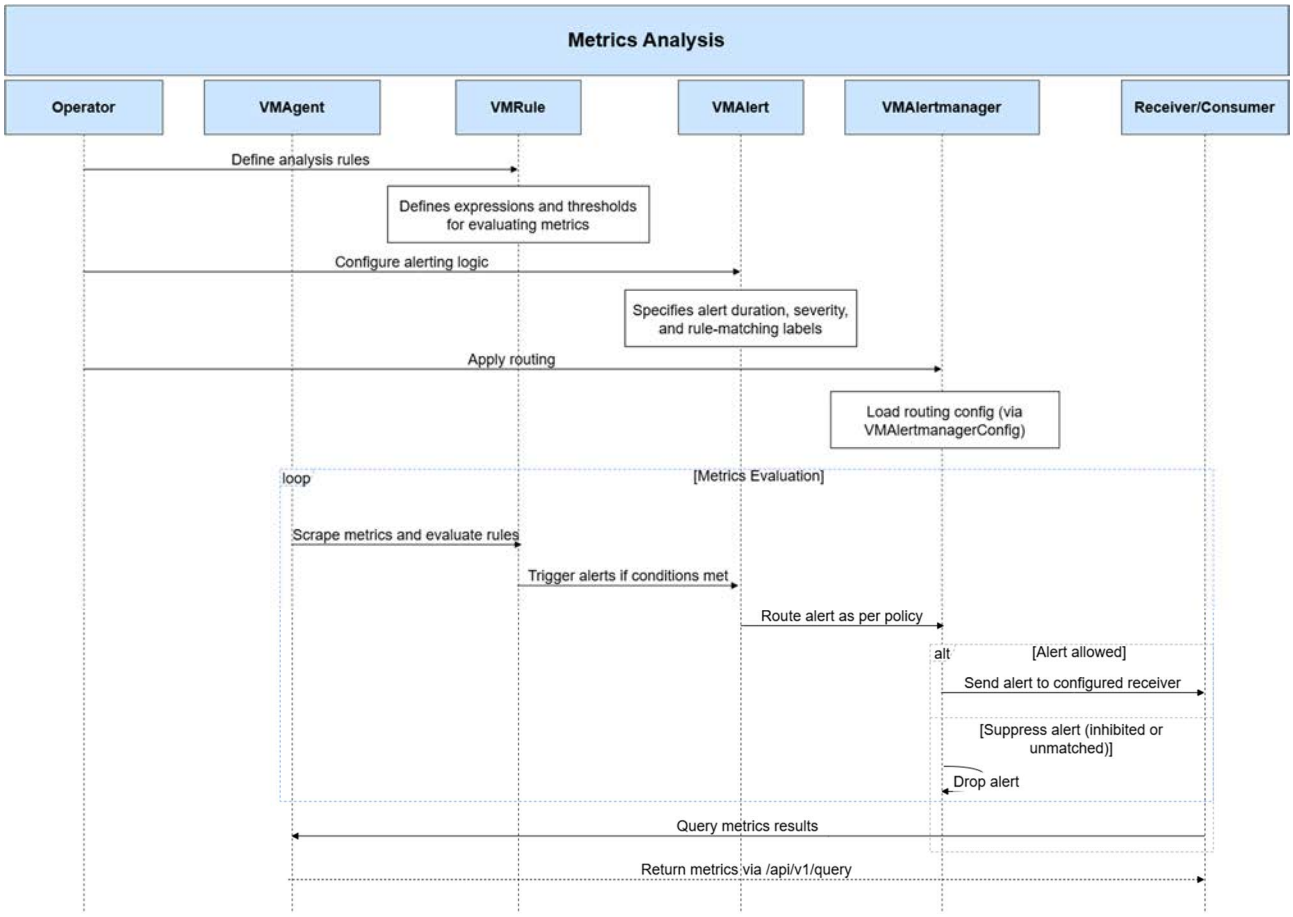

Figure 5.5-1: Metrics analysis with VictoriaMetrics using VMAgent, VMRule, VMAlert, VMAlertmanager

6 Traffic Management interface

6.1 Description

This interface allows the API consumer to invoke Istio® traffic management operations towards the API producer through the interface operations specified in clause 6.2.1 of ETSI GS NFV-IFA 049 [1], primarily focusing on traffic blocking and rerouting requests initiated by the API Consumer. Istio® DestinationRule, and AuthorizationPolicy custom resource objects are identified as Traffic Management related NFV object (Traffic Enforcer object) as defined in clause 5 of the present document.

The operations provided through this interface are:

- Create DestinationRule, and AuthorizationPolicy resources.

NOTE: The Kubernetes® API supports PUT, PATCH, and GET operations on Istio® resource objects; however, these operations are out of the scope of the present document. Additionally, the DELETE/Termination operation is not supported in this context, as the Traffic Enforcer function might remain available to ensure continuous traffic rerouting or isolation.

6.2 API version

The API {VERSION} for the profiled solution Istio® [11] custom resource object identified as Traffic Management related NFV object shall be set to "v1". Details on the API structure are specified in clause 4.3.2 of the present document.

The corresponding Istio® API roots are specified as:

/apis/networking.istio.io/v1

/apis/security.istio.io/v1

6.3 Resource structure and methods

Figures 6.3-1 and 6.3-2, show the overall resource URI structures for the profiled solution Istio® [11] for the Traffic Management interface.

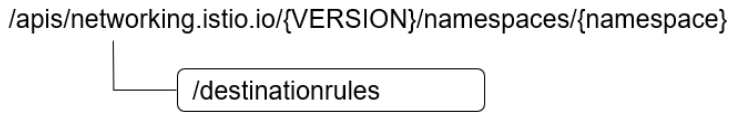

Figure 6.3-1: Resource URI structure of DestinationRule resource object for the Traffic Management interface

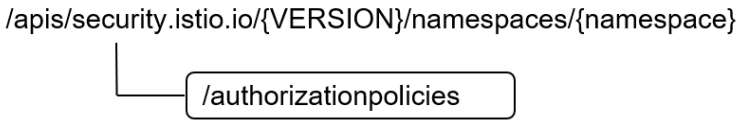

Figure 6.3-2: Resource URI structure of AuthorizationPolicy resource object for the Traffic Management interface

Table 6.3-1 lists the individual resources defined, and the applicable HTTP methods.

The Traffic Enforcer function supports responding to requests for all HTTP methods on the resources in Table 6.3-1 that are marked as "M" (mandatory) in the "Cat" column.

Table 6.3-1: Resources and methods overview of the Traffic Management interface

Resource name	Resource URI	HTTP Method	Cat	Meaning
DestinationRule	/destinationrules	POST	M	Create a new "DestinationRule" resource.
AuthorizationPolicy	/authorizationpolicies	POST	M	Create a new "AuthorizationPolicy" resource.

6.4 Sequence diagrams (informative)

See clause B.1 for the sequence diagrams for the Istio® resources.

6.5 Resources

6.5.1 Introduction

This clause profiles the resources and methods provided by the Traffic Management interface.

6.5.2 Resource: DestinationRule

This resource represents the Istio® [11] custom resource object DestinationRule, which specifies traffic policies, such as load balancing, connection pooling, and mutual TLS, for routing requests to the pods backing a service.

Table 6.5.2-1 provides the profiling of the supported DestinationRule resource methods against the Traffic Management interface requirements as specified in ETSI GS NFV-IFA 049 [1].

The URI query parameters, request and response bodies, and response codes of the individual resource methods are described in the respective DestinationRule custom resource object specifications of the profiled solution Istio®.

Table 6.5.2-1: DestinationRule resource methods profiling against Traffic Management interface requirements

Resource URI	HTTP Method	Meaning	Requirement identifier from ETSI GS NFV-IFA 049 [1]
/destinationrules	POST	Create a new "DestinationRule" resource.	TrafficEnf.Trafm.001

NOTE: Since interface requirement specified in clause 6.2.1 of ETSI GS NFV-IFA 049 [1] primarily focuses on traffic rerouting for VNFC instances, which can be fulfilled by the DestinationRule, provided the appropriate Gateway and VirtualService are created in advance.

6.5.3 Resource: AuthorizationPolicy

This resource represents the Istio® [11] custom resource object AuthorizationPolicy, which defines precise access control rules to allow or deny requests to workloads in the service mesh, based on criteria such as source, destination, and request attributes.

Table 6.5.3-1 provides the profiling of the supported AuthorizationPolicy resource methods against the Traffic Management interface requirements as specified in ETSI GS NFV-IFA 049 [1].

The URI query parameters, request and response bodies, and response codes of the individual resource methods are described in the respective AuthorizationPolicy custom resource object specifications of the profiled solution Istio®.

Table 6.5.3-1: AuthorizationPolicy resource methods profiling against Traffic Management interface requirements

Resource URI	HTTP Method	Meaning	Requirement identifier from ETSI GS NFV-IFA 049 [1]
/authorizationpolicies	POST	Create a new "AuthorizationPolicy" resource.	TrafficEnf.Trafm.001

NOTE: Since interface requirement specified in clause 6.2.1 of ETSI GS NFV-IFA 049 [1] primarily focuses on traffic blocking for VNFC instances, which can be fulfilled by the AuthorizationPolicy.

6.6 Data Model

6.6.1 Traffic Management operation input parameters mapped to CRD schemas configuration fields

This clause maps the Traffic Management operation input parameters as specified in clause 6.3.1.2.2 of ETSI GS NFV-IFA 049 [1] with the configuration fields of the example CRD schemas for DestinationRule and AuthorizationPolicy provided in clause A.5.3.

Table 6.6.1-1: Traffic Management operation input parameters mapped to Istio® CRD schemas configuration fields

Input parameter from ETSI GS NFV-IFA 049 [1]	Configuration Fields profiled from Istio® CRD Schemas	Related Istio® Resource	Description
vnfcInstanceId	subsets.labels	DestinationRule	One or more named sets that represent individual versions of a service. Traffic policies can be overridden at subset level. "subsets.labels" field identifies specific set of pods/VMs on which a policy should be applied.
targetAction	action	AuthorizationPolicy	The action to take if the request is matched with the rules. Default is ALLOW if not specified. Action specifies the operation to take. • ALLOW • DENY • AUDIT • CUSTOM "action" field can help in traffic isolation by denying traffic based on rules, such as blocking traffic from specific namespaces.
	rules	AuthorizationPolicy	A list of rules to match the request. A match occurs when at least one rule matches the request. Such as rules.from and rules.to fields, can specify different namespaces, paths, hosts, ports, etc. "rules" field can help enforce access control by specifying conditions for requests. By combining these rules with appropriate actions (such as ALLOW, DENY), it can lead to partial or full isolation by restricting access under specific conditions.
	selector.matchlabels	AuthorizationPolicy	One or more labels that indicate a specific set of pods/VMs on which a policy should be applied. The scope of label search is restricted to the configuration namespace in which the resource is present.

7 Log Exposure Interface

7.1 Description

This interface allows the API consumer to invoke Fluent Bit log aggregator operations towards the API producer through the interface operations specified in clause 6.2.4 of ETSI GS NFV-IFA 049 [1], primarily focusing on logs exposing and filtering requests initiated by the API Consumer. Fluent Bit ClusterFilter, and ClusterOutput custom resource objects are identified as Log Exposure related NFV object (Log Aggregator object) as defined in clause 5 of the present document.

The operations provided through this interface are:

- Create ClusterFilter, and ClusterOutput resources.

NOTE: The Kubernetes® API supports PUT, PATCH, and GET operations on Fluent Bit resource objects; however, these operations are out of the scope of the present document. Additionally, the DELETE/Termination operation is not supported in this context, as the Log Aggregator function might remain available to ensure continuous log aggregation and processing.

7.2 API version

The API {VERSION} for the profiled solution Fluent Bit [12] custom resource object identified as Log Exposure related NFV object shall be set to "v1alpha2". Details on the API structure are specified in clause 4.3.2 of the present document.

The corresponding Fluent Bit API roots are specified as:

/apis/fluentbit.fluent.io/v1alpha2

7.3 Resource structure and methods

Figures 7.3-1 and 7.3-2 show the overall resource URI structures for the profiled solution Fluent Bit [12] for the Log Exposure interface.

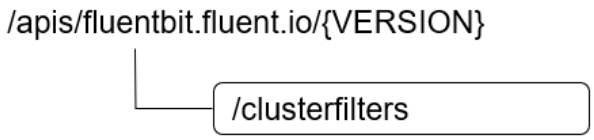

Figure 7.3-1: Resource URI structure of ClusterFilter resource object for the Log Exposure interface

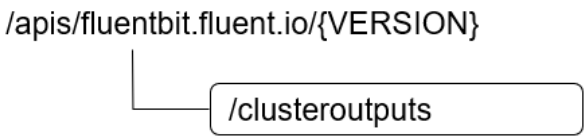

Figure 7.3-2: Resource URI structure of ClusterOutput resource object for the Log Exposure interface

Table 7.3-1 lists the individual resources defined, and the applicable HTTP methods.

The Log Aggregator function supports responding to requests for all HTTP methods on the resources in Table 7.3-1 that are marked as "M" (mandatory) in the "Cat" column.

Table 7.3-1: Resources and methods overview of the Log Exposure interface

Resource name	Resource URI	HTTP Method	Cat	Meaning
ClusterFilter	/clusterfilters	POST	M	Create a new "ClusterFilter" resource.
ClusterOutput	/clusteroutputs	POST	M	Create a new "ClusterOutput" resource.

7.4 Sequence diagrams (informative)

See clause B.2 for the sequence diagrams for the Fluent Bit resources.

7.5 Resources

7.5.1 Introduction

This clause profiles the resources and methods provided by the Log Exposure interface.

7.5.2 Resource: ClusterFilter

This resource represents the Fluent Bit [12] custom resource object ClusterFilter, which modifies, enriches, or excludes log data before output.

Table 7.5.2-1 provides the profiling of the supported ClusterFilter resource methods against the Log Exposure interface requirements as specified in ETSI GS NFV-IFA 049 [1].

The URI query parameters, request and response bodies, and response codes of the individual resource methods are described in the respective ClusterFilter custom resource object specifications of the profiled solution Fluent Bit.

Table 7.5.2-1: ClusterFilter resource methods profiling against Log Exposure interface requirements

Resource URI	HTTP Method	Meaning	Requirement identifier from ETSI GS NFV-IFA 049 [1]
/clusterfilters	POST	Create a new "ClusterFilter" resource.	LogAggr.Expose.002

7.5.3 Resource: ClusterOutput

This resource represents the Fluent Bit [12] custom resource object ClusterOutput, which defines where the processed log data is sent.

Table 7.5.3-1 provides the profiling of the supported ClusterOutput resource methods against the Log Exposure interface requirements as specified in ETSI GS NFV-IFA 049 [1].

The URI query parameters, request and response bodies, and response codes of the individual resource methods are described in the respective ClusterOutput custom resource object specifications of the profiled solution Fluent Bit.

Table 7.5.3-1: ClusterOutput resource methods profiling against Log Exposure interface requirements

Resource URI	HTTP Method	Meaning	Requirement identifier from ETSI GS NFV-IFA 049 [1]
/clusteroutputs	POST	Create a new "ClusterOutput" resource.	LogAggr.Expose.001 See note.
NOTE: The requirement LogAggr.Expose.001 from ETSI GS NFV-IFA 049 [1] specifies that the Log Exposure Interface shall support exposing the logs to authorized consumers. However, in the context of "ClusterOutput," the focus is on transmitting logs to multiple destinations for purposes such as analysis, storage, or alerting. Most existing log aggregation solutions are designed to collect and distribute logs across various endpoints for these purposes, including visualization. While the "ClusterOutput" resource contributes to fulfilling this requirement, it does so in an indirect manner by facilitating log distribution rather than direct exposure.			

7.6 Data model

7.6.1 Log Exposure operation input parameters mapped to CRD schemas configuration fields

This clause maps the Log Exposure operation input parameters as specified in clause 6.3.4.2.2 of ETSI GS NFV-IFA 049 [1] with the configuration fields of the example CRD schemas for ClusterFilter and ClusterOutput provided in clause A.5.2.

Table 7.6.1-1: Log Exposure operation input parameters mapped to Fluent Bit CRD schemas configuration fields

Input parameter from ETSI GS NFV-IFA 049 [1]	Configuration Fields profiled from Fluent Bit CRD Schemas	Related Fluent Bit Resource	Description
Filter	filters.grep	ClusterFilter	The filters grep field is to processes logs by matching or excluding patterns using regular expressions. It supports options like regex for keeping logs that match specific fields and exclude for filtering out unwanted entries, along with other configurations.
	filters.kubernetes	ClusterFilter	The filters.kubernetes® field is to add Kubernetes® metadata to logs, such as pod names, namespaces, labels, and annotations. It includes options like kubeURL for accessing the API server, mergeLog to include JSON log content, and regexParser for custom tag parsing for Kubernetes-based environments.
	filters.modify	ClusterFilter	The filters.modify field allows dynamic changes to log data, applying rules like add to insert new fields, remove to delete fields, rename to change field names, and more. It supports conditions for checking key matches and other operations for various log manipulation needs.
	filters.parser	ClusterFilter	The filters.parser field extracts and structures data from log fields using predefined or custom parsers. Features like preserveKey to keep original fields and reserveData to include unprocessed fields make it useful for working with different log formats, such as JSON or custom structures.
Not specified	file	ClusterOutput	The file field is to configure Fluent Bit to output logs to files in various formats. It supports options like delimiter for CSV or LTSV formats, file for setting the filename, path for defining the directory, and format for the file structure (e.g. out_file, csv, template). The template option allows for custom formatting, and many other configuration options are also available for further customization.
	opentelemetry	ClusterOutput	The Opentelemetry™ field is to configure Fluent Bit to export logs and metrics to OpenTelemetry™compatible systems. It supports adding custom labels (addLabel), specifying HTTP headers (header), and defining the target server with host and port. Additionally, it provides networking options such as DNS settings and source address binding for connectivity. Many other tags are also available to send logs to different destinations, with additional configuration options for further customization.

NOTE: According to clause 6.3.4.2 of ETSI GS NFV-IFA 049 [1], the input parameter Filter is used to retrieve logs as an output. In the context of Fluent Bit, however, the ClusterFilter custom resource processes, transforms, and structures logs, enabling the filtered and processed logs to be sent to various destinations for analysis, storage, or alerting. The ClusterOutput custom resource defines these destinations and ensures log delivery. Need to check with ETSI GS NFV-IFA 049 [1] to add a new input parameter as "Output" or "Destination" to expose the logs.

8 Log Analysis Exposure Interface

8.1 Description

This interface allows the API consumer to invoke OpenSearch operations towards the API producer through the interface operations specified in clause 6.2.5 of ETSI GS NFV-IFA 049 [1]. The interface supports both OpenSearch resource management and log retrieval operations:

- The OpenSearchCluster custom resource object is identified as a Log Analysis Exposure related NFV object (Log Analyser object), as defined in clause 5.3 of the present document. It provides a declarative way to manage OpenSearch clusters and configurations.
- The PPL API enables consumers to retrieve and analyse log data dynamically. PPL allows structured queries for filtering, sorting, and aggregating logs without modifying OpenSearch resources. See clause 4.3.4.2 in the present document for details.

The operations provided through this interface are:

- Create OpenSearchCluster resource.
- Retrieve the log data.

NOTE: The Kubernetes® API supports PUT, PATCH, and GET operations on OpenSearch resource objects; however, these operations are out of the scope of the present document. Additionally, the DELETE/Termination operation is not supported in this context, as the Log Analyser function might remain available to continue providing log-based analytics to authorised consumers.

8.2 API version

The API {VERSION} for the profiled solution OpenSearch [13] custom resource object identified as Log Analysis Exposure related NFV object shall be set to "v1". Details on the API structure are specified in clause 4.3.2 of the present document.

The corresponding OpenSearch API roots are specified as:

/apis/opensearch.opster.io/v1

The API {version} for the PPL API shall be set to "1.0.0". The PPL API does not expose a versioned endpoint in the URI; instead, the API version is specified in the OpenAPI metadata under the info.version field.

8.3 Resource structure and methods

Figure 8.3-1, show the overall resource URI structures for the profiled solution OpenSearch [13] for the Log Analysis Exposure interface.

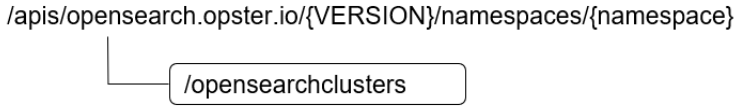

Figure 8.3-1: Resource URI structure of OpenSearchCluster resource object for the Log Analysis Exposure interface

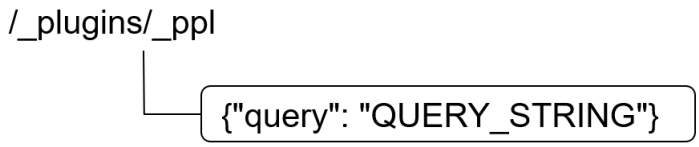

Figure 8.3-2: Resource URI structure of PPL API

Table 8.3-1 lists the individual resources defined, and the applicable HTTP methods.

The Log Analyser function supports responding to requests for all HTTP methods on the resources in Table 8.3-1 that are marked as "M" (mandatory) in the "Cat" column.

Table 8.3-1: Resources and methods overview of the Log Analysis Exposure interface

Resource name	Resource URI	HTTP Method	Cat	Meaning
OpenSearchCluster	/opensearchclusters	POST	M	Create a new "/OpenSearchCluster" resource.
PPL API	/_plugins/_ppl	POST	M	Retrieve the log data.

8.4 Sequence diagrams (informative)

See clause B.3 for the sequence diagrams for the OpenSearch resources.

8.5 Resources

8.5.1 Introduction

This clause profiles the resources and methods provided by the Log Analysis Exposure interface.

8.5.2 Resource: OpenSearchCluster

This resource represents the OpenSearch [13] custom resource object OpenSearchCluster, which manages the deployment and configuration of an OpenSearch cluster including OpenSearch dashboards.

Table 8.5.2-1 provides the profiling of the supported OpenSearchCluster resource methods against the Log Analysis Exposure interface requirements as specified in ETSI GS NFV-IFA 049 [1].

The URI query parameters, request and response bodies, and response codes of the individual resource methods are described in the respective OpenSearchCluster custom resource object specifications of the profiled solution OpenSearch.

Table 8.5.2-1: OpenSearchCluster resource methods profiling against Log Analysis Exposure interface requirements

Resource URI	HTTP Method	Meaning	Requirement identifier from ETSI GS NFV-IFA 049 [1]
/opensearchclusters	POST	Create a new "OpenSearchCluster" resource.	LogAnalyser.Expose.002

8.5.3 Resource: PPL API

This resource represents the OpenSearch PPL API [15], to retrieve the log data dynamically based on query parameters.

The PPL query operation is performed by sending an HTTP POST request to the /_plugins/_ppl endpoint, with a JSON-based request body containing a valid PPL query. The response returns a structured JSON object containing the query results.

Table 8.5.3-1 provides details of the PPL API operation.

Table 8.5.3-1: PPL API profiling against Log Analysis Exposure interface requirements

Resource URI	HTTP Method	Meaning	Requirement Identifier
/_plugins/_ppl {"query": "QUERY_STRING"}	POST	Execute a PPL query for structured log retrieval.	LogAnalyser.Expose.001

8.6 Data model

8.6.1 Log Analysis Exposure operation input parameters mapping

This clause maps the Log Analysis Exposure operation input parameters as specified in clause 6.3.5.2.2 of ETSI GS NFV-IFA 049 [1] with the configuration fields in the example OpenSearchCluster CRD schema provided in clause A.5.1, and the PPL API endpoint.

Table 8.6.1-1: Log Analysis Exposure operation input parameters mapped to OpenSearch CRD schemas configuration fields

Input parameter from ETSI GS NFV-IFA 049 [1]	Configuration Fields profiled from OpenSearch CRD Schemas	Related OpenSearch Resource	Description
LogAnalysisConfig	Not available	OpenSearchCluster	The OpenSearchCluster CRD schema does not contain a corresponding configuration field for the input parameter 'LogAnalysisConfig'. As stated in clause 5.3 of the present document, OpenSearchCluster provisions an OpenSearch cluster along with dashboards. These dashboards provide an interface for performing log filtering, applying log analysis functions, and aggregating log data, ensuring efficient log analysis and visualization. Therefore, the requirements for log analysis can be fulfilled using the OpenSearch dashboards provided by the OpenSearchCluster custom resource.

Table 8.6.1-2 details how the PPL API (/_plugins/_ppl) supports log filtering in the Log Analysis Exposure interface.

Table 8.6.1-2: Log Analysis Exposure operation input parameters mapped to PPL API endpoint

Input parameter from ETSI GS NFV-IFA 049 [1]	Related OpenSearch API Endpoint	Description
Filter	PPL API endpoint (/_plugins/_ppl)	The PPL API allows dynamic log retrieval and filtering by enabling structured queries that specify conditions for log selection. API consumers can apply filtering criteria using PPL query syntax, such as the WHERE clause, to extract only relevant log entries based on predefined conditions.

9 Interfaces exposed by the PaaS Service Notification Manager

9.1 Description

This interface allows the API consumer to invoke Prometheus Alertmamanger operations towards the API producer through the interface operations specified in clause 6.2.11 of ETSI GS NFV-IFA 049 [1], primarily focusing on managing alert rules and notification configurations requests initiated by the API Consumer. Prometheus AlertmanagerConfig and Alertmanager custom resource objects are identified as Notification manager related NFV object (Prometheus Alertmanager object) as defined in clause 5 of the present document.

The operations provided through this interface are:

- Create AlertmanagerConfig and Alertmanager resources

NOTE: The Kubernetes® API supports POST, PUT, PATCH, and GET operations on Prometheus Alertmanger resource objects; however, these operations are out of the scope of the present document. Additionally, the DELETE/Termination operation is not supported in this context, as the Notification Manager function might remain available to process and deliver notifications to users.

9.2 API version

The API {VERSION} for the profiled solution Prometheus Alertmanager [14] custom resource object identified as Notification manager related NFV object shall be set to " v1alpha1" for AlertmanagerConfig and "v1" for Alertmanager CRD object. Details on the API structure are specified in clause 4.3.2 of the present document.

The corresponding Prometheus Alertmanager API roots are specified as:

/api/ monitoring.coreos.com/v1alpha1,

/api/ monitoring.coreos.com/v1

9.3 Resource structure and methods

Figures 9.3-1 and 9.3-2 show the overall resource URI structures for the profiled solution Prometheus Alertmanager [14] for the Alert and Receiver interfaces:

/apis/monitoring.coreos.com/v1alpha1/AlertmanagerConfig,

/apis/monitoring.coreos.com/v1/Alertmanager

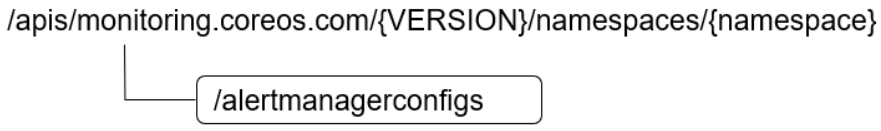

Figure 9.3-1: Resource URI structure of AlertmanagerConfig resource object for the Notification Manager interface

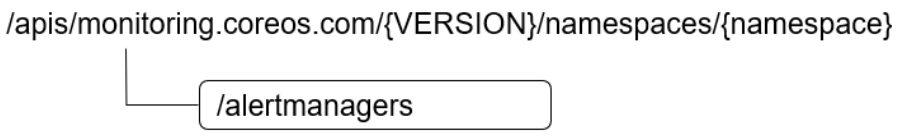

Figure 9.3-2: Resource URI structure of Alertmanager resource object for the Notification Manager interface

Table 9.3-1 lists the individual resources defined, and the applicable HTTP methods.

The notification manager function supports responding to requests for all HTTP methods on the resources in Table 9.3-1 that are marked as "M" (mandatory) in the "Cat" column.

Table 9.3-1: Resources and Methods Overview of the Prometheus Alertmanager Interface

Resource name	Resource URI	HTTP Method	Cat	Meaning
AlertmanagerConfig	/alertmanagerconfigs	POST	M	Create a new "AlertmanagerConfig" resource.
Alertmanager	/alertmanagers	POST	M	Create a new "Alertmanager" resource.

9.4 Sequence diagrams (informative)

See clause B.4 for the sequence diagrams for the Prometheus Alertmanager resources.

9.5 Resources

9.5.1 Introduction

This clause profiles the resources and methods provided by the Prometheus Alertmanager interface.

9.5.2 Resource: AlertmanagerConfig

This resource represents the Prometheus Alertmanager [14] custom resource object AlertmanagerConfig, which specifies alert rules for monitoring metrics and generating alerts based on thresholds or conditions.

Table 9.5.2-1 provides the profiling of the supported AlertmanagerConfig resource methods against the Notification Manager interface requirements as specified in ETSI GS NFV-IFA 049 [1].

The URI query parameters, request and response bodies, and response codes of the individual resource methods are described in the respective AlertmanagerConfig custom resource object specifications of the profiled solution Prometheus Alertmanager.

Table 9.5.2-1: AlertmanagerConfig resource methods profiling against Notification Manager interface requirements

Resource URI	HTTP Method	Meaning	Requirement identifier from ETSI GS NFV-IFA 049 [1]
/alertmanagerconfigs	POST	Create a new "AlertmanagerConfig" resource.	Notif.Manager Notif.Mgmt.002

NOTE: Since interface requirements specified in clause 6.2.11 of ETSI GS NFV-IFA 049 [1] primarily focuses on sending processed notifications to authorized consumers, this can be fulfilled by the AlertmanagerConfig, which enables the definition of rules and configurations for sending notifications through Alertmanager.

9.5.3 Resource: Alertmanager

This resource represents the Prometheus Alertmanager [14] custom resource object Alertmanager, which defines the configuration for the deployment of the Alertmanager instance, alert routing logic (routing policies to determine how alerts should be handled), specifying storage to ensure alert history and sending notifications to defined receivers.

Table 9.5.3-1 provides the profiling of the supported Alertmanager resource methods against the Notification Manager interface requirements as specified in ETSI GS NFV-IFA 049 [1].

The URI query parameters, request and response bodies, and response codes of the individual resource methods are described in the respective Alertmanager custom resource object specifications of the profiled solution Prometheus Alertmanager.

Table 9.5.3-1: Alertmanager resource methods profiling against Notification Manager interface requirements

Resource URI	HTTP Method	Meaning	Requirement identifier from ETSI GS NFV-IFA 049 [1]
/alertmanagers	POST	Create a new "Alertmanager" resource.	Notif.Manager Notif.Mgmt.002

NOTE: Since interface requirements specified in clause 6.2.11 of ETSI GS NFV-IFA 049 [1] primarily focuses on sending processed notifications to authorized consumers, this can be fulfilled by the Alertmanager, which enables the define alerting routing logic for sending notifications to defined receivers.

9.6 Data Model

9.6.1 Notification Manager operation input parameters mapped to CRD schemas configuration fields

This clause maps the Notification Manager operation input parameters as specified in clause 6.3.12.2.2 of ETSI GS NFV-IFA 049 [1] with the configuration fields of the example CRD schemas for AlertmanagerConfig and Alertmanager provided in clause A.5.4.

Table 9.6.1-1: Notification Management operation input parameters mapped to Prometheus CRD schemas configuration fields

Input parameter from ETSI GS NFV-IFA 049 [1]	Configuration Fields profiled from Prometheus Alertmanager CRD Schemas	Related Prometheus Alertmanager Resource	Description
filter	receivers	AlertmanagerConfig	Defines the type of receiver (e.g. email, webhook).
	receivers.emailConfigs	AlertmanagerConfig	Specifies email addresses for sending notifications.
	route	AlertmanagerConfig	Specifies grouped and routed definitions for alerts matching the resource's namespace.
	route.groupBy	AlertmanagerConfig	Groups alert by labels for efficient routing and processing.
	route.matchers	AlertmanagerConfig	Filters alerts based on label matchers to decide which alerts are handled by the receiver.
	route.receiver	AlertmanagerConfig	Directs alerts to specific receivers based on the alert priority.
	inhibitRules	AlertmanagerConfig	Prevents notifications for specific alerts if other defined alerts are already firing.
Not specified	storage	Alertmanager	Defines the storage configuration for Alertmanager, used for managing notification history.
Not specified	alertmanagerConfigSelector	Alertmanager	Dynamically manage notification routing and inhibition rules using AlertmanagerConfig CRD.

10 Metrics Analysis Exposure Interface

10.1 Description

This interface allows the API consumer to invoke VictoriaMetrics operations towards the API producer through the interface operations specified in clause 6.2.7 of ETSI GS NFV-IFA 049 [1]. The interface supports both VictoriaMetrics resource management and metrics retrieval operations:

- The VMRule custom resource object is identified as a Metrics Analysis Exposure related NFV object (Metrics Analyser object), as defined in clause 5.5 of the present document. It provides a declarative way to manage metrics analysis and alerting rules.
- The Query API is a read-only API, that allows consumers to dynamically retrieve and analyse metrics using MetricsQL for filtering and aggregation. See clause 4.3.4.3 in the present document for details.

The operations provided through this interface are:

- Create VMRule resource.
- Retrieve time-series metrics data.

NOTE: The Kubernetes® API supports PUT, PATCH, and GET operations on VictoriaMetrics resource objects; however, these operations are out of the scope of the present document. Additionally, the DELETE/Termination operation is not supported in this context, as the Metrics Analyser function might remain available to continue providing metrics-based analytics to authorised consumers.

10.2 API version

The API {VERSION} for the profiled solution VictoriaMetrics [9] custom resource object identified as Metrics Analysis Exposure related NFV object shall be set to "v1beta1". Details on the API structure are specified in clause 4.3.2 of the present document.

The corresponding VictoriaMetrics API roots are specified as:

/apis/operator.victoriametrics.com/v1beta1

The API {VERSION} for the Query API shall be set to "v1", as indicated by the "/api/v1/" path prefix. The version is defined by the path itself.

10.3 Resource structure and methods

Figure 10.3-1, show the overall resource URI structures for the profiled solution VictoriaMetrics [9] for the Metrics Analysis Exposure interface.

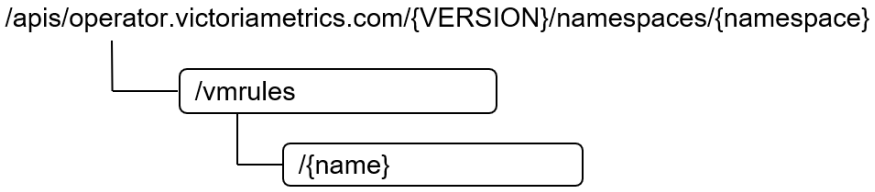

Figure 10.3-1: Resource URI structure of VMRule resource object for the Metrics Analysis Exposure interface

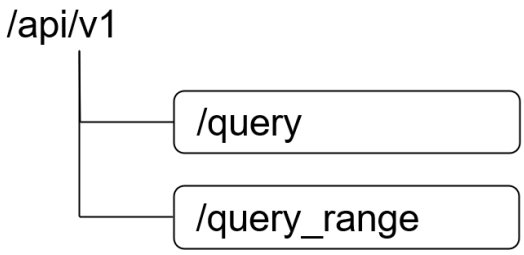

Figure 10.3-2: Resource URI structure of Query API

Table 10.3-1 lists the individual resources defined, and the applicable HTTP methods.

The Metrics Analyser function supports responding to requests for all HTTP methods on the resources in Table 10.3-1 that are marked as "M" (mandatory) in the "Cat" column.

Table 10.3-1: Resources and methods overview of the Metrics Analysis Exposure interface

Resource name	Resource URI	HTTP Method	Cat	Meaning
VMRule	/vmrules/{name}	POST	M	Create a new "VMRule" resource.
Query API	/query	POST	M	Retrieve time-series metrics data at a specific timestamp.
		GET	M	
	/query_range	POST	M	Retrieve time-series metrics data over a time range.
		GET	M	

10.4 Sequence diagrams (informative)

See clause B.5 for the sequence diagrams for the VictoriaMetrics resources.

10.5 Resources

10.5.1 Introduction

This clause profiles the resources and methods provided by the Metrics Analysis Exposure interface.

10.5.2 Resource: VMRule

This resource represents the VictoriaMetrics [9] custom resource object VMRule, which defines alerting and recording rules for evaluating metrics data stored in VictoriaMetrics.

Table 10.5.2-1 provides the profiling of the supported VMRule resource methods against the Metrics Analysis Exposure interface requirements as specified in clause 6.2.7 of ETSI GS NFV-IFA 049 [1].

The URI query parameters, request and response bodies, and response codes of the individual resource methods are described in the respective VMRule custom resource object specifications of the profiled solution VictoriaMetrics.

Table 10.5.2-1: VMRule resource methods profiling against Metrics Analysis Exposure interface requirements

Resource URI	HTTP Method	Meaning	Requirement identifier from ETSI GS NFV-IFA 049 [1]
/VMRules/{name}	POST	Create a new "VMRule" resource.	MetricAnalyser.Expose.002

10.5.3 Resource: Query API

This resource represents the VictoriaMetrics Query API [10], used to retrieve time-series metrics data dynamically based on query parameters.

The query operations can be performed by sending either an HTTP GET or POST request to the /api/v1/query or /api/v1/query_range endpoints, with query parameters provided in the URL or as URL-encoded form data (in key=value pairs) in the request body, following Prometheus® API-compatible behaviour. The response returns a structured JSON object containing the query results.

Table 10.5.3-1 provides details of the Query API operations.

Table 10.5.3-1: Query API profiling against Metrics Analysis Exposure interface requirements

Resource URI	HTTP Method	Meaning	Requirement identifier from ETSI GS NFV-IFA 049 [1]
/api/v1/query	POST	Execute a query to retrieve the value at a specific timestamp, with parameters provided in request body as URL-encoded form data.	MetricAnalyser.Expose.001
	GET	Execute a query to retrieve the value at a specific timestamp, with parameters provided in URL.	
/api/v1/query_range	POST	Execute a query to retrieve values over a specified time range, with parameters provided in request body as URL-encoded form data.	
	GET	Execute a query to retrieve values over a specified time range with parameters provided in URL.	

10.6 Data model

10.6.1 Metrics Analysis Exposure operation input parameters mapping

This clause maps the Metrics Analysis Exposure operation input parameters as specified in clause 6.3.7.2 of ETSI GS NFV-IFA 049 [1] with the configuration fields in the example VMRule CRD schema provided in clause A.5.5, and the query API endpoints.

Table 10.6.1-1: Metrics Analysis Exposure operation input parameters mapped to VictoriaMetrics CRD schemas configuration fields

Input parameter from ETSI GS NFV-IFA 049 [1]	Configuration Fields profiled from VictoriaMetrics CRD Schemas	Related VictoriaMetrics Resource	Description
MetricsAnalysisConfig	groups[].interval	VMRule	Defines how often the rule group should be evaluated. It controls the frequency of metric analysis execution.
	groups[].rules[].expr	VMRule	The MetricsQL expression that specifies the logic for analyzing the collected metrics. It determines what conditions or patterns are checked.
	groups[].rules[].alert	VMRule	The name of the alert to be triggered if the specified `expr` condition is true. It marks the result as a named alert for further handling.
	groups[].rules[].for	VMRule	Specifies the minimum duration the condition shall remain true before the alert is triggered. It helps reduce false positives by requiring sustained conditions.
	groups[].rules[].labels	VMRule	Custom labels attached to the alert or recorded result for categorization, filtering, or routing. It is useful for tagging results with metadata like severity or environment.
	groups[].rules[].record	VMRule	Defines a new metric name for storing the result of the expression as a time-series metric, enabling reuse in dashboards or further analysis.

Table 10.6.1-2 details how the query API (/api/v1/query, /api/v1/query_range) supports metrics filtering in the Metrics Analysis Exposure interface.

Table 10.6.1-2: Metrics Analysis Exposure operation input parameters mapped to query API endpoints

Input parameter from ETSI GS NFV-IFA 049 [1]	Related VictoriaMetrics API Endpoints	Description
Filter	Query API endpoints (/api/v1/query, /api/v1/query_range)	Provides a way to retrieve, filter, and aggregate time-series metrics data using MetricsQL queries over HTTP, enabling real-time analysis of metrics stored in VictoriaMetrics.

Annex A (informative): Analysis on the existing solutions based on the interfaces exposed by the VNF generic OAM functions and other PaaS Services

A.1 Comparison of the VNF generic OAM functions and other PaaS Services functional requirements with cloud native open source solutions

A.1.1 Overview

This clause analyses comparison of the VNF generic OAM functions and other PaaS Services functional requirements specified in ETSI GS NFV-IFA 049 [1] and open source solutions that fit in CNCF and LFN landscapes.

A.1.2 Comparison of Log Aggregator functional requirements with relevant open-source solutions capabilities

A.1.2.1 Fluent Bit

A.1.2.1.1 Overview

This clause analyses the comparison of Log Aggregator functional requirements defined in clause 5.5 of ETSI GS NFV IFA 049 [1], and the open-source solution Fluent Bit.

Fluent Bit is an open-source tool that collects logs and metrics from multiple sources, primarily for log-centric use cases. It enables data enrichment through filters, parsing, and efficiently routes the processed data to defined destinations. It has been developed with a focus on performance to allow the collection and processing of telemetry data from different sources without complexity.

A.1.2.1.2 Comparison

This clause shows comparison of Log Aggregator functional requirements defined in clause 5.5 of ETSI GS NFV-IFA 049 [1] (see "Identifier" column and "Requirement" column in Table A.1.2.1.2-1) and Fluent Bit (see "Support by open source" and "Related capability of open source" column). The legend of "Support by open source" is the following:

"Yes": fully support the functional requirements.

"No": not support the functional requirements.

"Partial": partial support the functional requirements.

NOTE: The text reproduced in tables in clauses A.1 and A.2 was extracted from clause 5.5 of ETSI GS NFV IFA 049 [1] for readability purposes. Requirements reproduced in these tables are to be considered as quotes as they are not new requirements.

Table A.1.2.1.2-1: Comparison of Log Aggregator functional requirements and Fluent Bit

Identifier	Requirement	Support by open source	Related capability of open source
LogAggregator.001	The Log Aggregator function shall support the capability to collect different types of logs from different entities like the VNF instances (including the VNF's applications), NFV-MANO or NFVI (compute, storage, network resources) determined by a filter (see note 1).	Yes	Collect logs from various sources including VNF instances, applications, containers, and infrastructure components. It supports flexible filtering based on log attributes, source types, and metadata.
LogAggregator.002	The Log Aggregator function shall support the capability to pre-process the logs (see note 2).	Yes	Provides log parsing and format conversion capabilities. It can harmonize log formats using built-in parsers or custom ones and perform basic enrichment like adding metadata.
LogAggregator.003	The Log Aggregator function shall support the capability to aggregate the logs in a configurable manner (see note 3).	Yes	Supports log aggregation and routing based on configurable conditions. It can group logs based on criteria such as log level, source, or other attributes and forward them to appropriate destinations.
LogAggregator.004	The Log Aggregator function shall support the capability to store historical log records (see note 4).	Yes	It forwards logs to external storage systems for long-term storage.
LogAggregator.005	The Log Aggregator function shall support the capability to expose (filtered) logs to authorized consumers.	Yes	It forwards filtered logs to external systems that provide capabilities to expose logs to authorized consumers through dashboards or APIs.
NOTE 1: As an example for the case of VNF/VNFC instances, the filter shall support filtering of VNF/VNFC instances by type of the VNF/VNFC, vendor, host, zone, VNF instance identifier, etc. Also, it shall be able to filter by log attributes metric/log type, severity level, etc. NOTE 2: One form of pre-processing is to harmonize the format of the logs. NOTE 3: Examples of configurable forms of aggregation are to aggregate all logs based on criteria of log level, different instances belonging to the same VNF, VNF instances managed by the same VNFM, etc. NOTE 4: A use case to store historical log records is about using such records for further root-cause analysis.			

A.1.2.2 Fluentd

A.1.2.2.1 Overview

This clause analyses the comparison of Log Aggregator functional requirements defined in clause 5.5 of ETSI GS NFV IFA 049 [1], and the open-source solution Fluentd.

Fluentd is an open-source log collector that unifies data collection and routing, enabling efficient gathering, processing, and forwarding of log data from various sources to multiple destinations.

A.1.2.2.2 Comparison

This clause shows comparison of Log Aggregator functional requirements defined in clause 5.5 of ETSI GS NFV IFA 049 [1] (see "Identifier" column and "Requirement" column in Table A.1.2.2.2-1) and Fluentd (see "Support by open source" and "Related capability of open source" column). The legend of "Support by open source" is the following:

"Yes": fully support the functional requirements.

"No": not support the functional requirements.

"Partial": partial support the functional requirements.

Table A.1.2.2.2-1: Comparison of Log Aggregator functional requirements and Fluentd

Identifier	Requirement	Support by open source	Related capability of open source
LogAggregator.001	The Log Aggregator function shall support the capability to collect different types of logs from different entities like the VNF instances (including the VNF's applications), NFV-MANO or NFVI (compute, storage, network resources) determined by a filter (see note 1).	Yes	Collect logs from various sources including VNF instances, applications, containers, and infrastructure components. It supports flexible filtering based on log attributes, source types, and metadata.
LogAggregator.002	The Log Aggregator function shall support the capability to pre-process the logs (see note 2).	Yes	Provides filtering and parsing plugins to standardize log formats and modify log records as needed.
LogAggregator.003	The Log Aggregator function shall support the capability to aggregate the logs in a configurable manner (see note 3).	Yes	Offers output plugins for aggregation based on various criteria (e.g. log level, instance type) and supports buffering for efficient log handling.
LogAggregator.004	The Log Aggregator function shall support the capability to store historical log records (see note 4).	Yes	It forwards logs to external systems or vendor-specific backend that manage long-term storage and historical retention.
LogAggregator.005	The Log Aggregator function shall support the capability to expose (filtered) logs to authorized consumers.	Yes	It forwards filtered logs to external systems that provide capabilities to expose logs to authorized consumers through dashboards or APIs.
NOTE 1: As an example for the case of VNF/VNFC instances, the filter shall support filtering of VNF/VNFC instances by type of the VNF/VNFC, vendor, host, zone, VNF instance identifier, etc. Also, it shall be able to filter by log attributes metric/log type, severity level, etc. NOTE 2: One form of pre-processing is to harmonize the format of the logs. NOTE 3: Examples of configurable forms of aggregation are to aggregate all logs based on criteria of log level, different instances belonging to the same VNF, VNF instances managed by the same VNFM, etc. NOTE 4: A use case to store historical log records is about using such records for further root-cause analysis.			

A.1.2.3 OpenTelemetry Collector

A.1.2.3.1 Overview

This clause analyses the comparison of Log Aggregator functional requirements defined in clause 5.5 of ETSI GS NFV IFA 049 [1], and the open-source solution OpenTelemetry Collector.OpenTelemetry Collector is an open-source tool for collecting, processing, and exporting telemetry data (logs, metrics, traces) from various sources. It also provides a flexible pipeline for observability data routing to multiple backends.

A.1.2.3.2 Comparison

This clause shows comparison of Log Aggregator functional requirements defined in clause 5.5 of ETSI GS NFV IFA 049 [1] (see "Identifier" column and "Requirement" column in Table A.1.2.3.2-1) and OpenTelemetry Collector (see "Support by open source" and "Related capability of open source" column). The legend of "Support by open source" is the following:

"Yes": fully support the functional requirements.

"No": not support the functional requirements.

"Partial": partial support the functional requirements.

Table A.1.2.3.2-1: Comparison of Log Aggregator functional requirements and OpenTelemetry Collector

Identifier	Requirement	Support by open source	Related capability of open source
LogAggregator.001	The Log Aggregator function shall support the capability to collect different types of logs from different entities like the VNF instances (including the VNF's applications), NFV-MANO or NFVI (compute, storage, network resources) determined by a filter (see note 1).	Yes	Collect logs from various sources, including VNF instances and other infrastructure components. It supports filtering logs through its processor components, allowing for customized log handling.
LogAggregator.002	The Log Aggregator function shall support the capability to pre-process the logs (see note 2).	Yes	Can pre-process logs using processors. It can perform operations such as format conversion, enrichment, and harmonization to standardize logs before forwarding them to their destinations.
LogAggregator.003	The Log Aggregator function shall support the capability to aggregate the logs in a configurable manner (see note 3).	Yes	It enables configurable log aggregation based on criteria like log level, source, VNF instances, or custom attributes using flexible pipelines.
LogAggregator.004	The Log Aggregator function shall support the capability to store historical log records (see note 4).	Yes	It forwards logs to external systems that manage long-term storage and historical retention, such as Prometheus® or vendor-specific backend.
LogAggregator.005	The Log Aggregator function shall support the capability to expose (filtered) logs to authorized consumers.	Yes	It forwards logs to systems that provide access controls and exposure functionalities, enabling the exposure of logs to authorized consumers through dashboards or APIs.
NOTE 1: As an example for the case of VNF/VNFC instances, the filter shall support filtering of VNF/VNFC instances by type of the VNF/VNFC, vendor, host, zone, VNF instance identifier, etc. Also, it shall be able to filter by log attributes metric/log type, severity level, etc. NOTE 2: One form of pre-processing is to harmonize the format of the logs. NOTE 3: Examples of configurable forms of aggregation are to aggregate all logs based on criteria of log level, different instances belonging to the same VNF, VNF instances managed by the same VNFM, etc. NOTE 4: A use case to store historical log records is about using such records for further root-cause analysis.			

A.1.2.4 Grafana Loki

A.1.2.4.1 Overview

This clause analyses the comparison of Log Aggregator functional requirements defined in clause 5.5 of ETSI GS NFV IFA 049 [1], and the open-source solution Grafana Loki.Grafana Loki is an open-source log aggregation system that efficiently stores and indexes log metadata, allowing for cost-effective log querying. It integrates with Grafana® for seamless visualization of logs.

A.1.2.4.2 Comparison

This clause shows comparison of Log Aggregator functional requirements defined in clause 5.5 of ETSI GS NFV IFA 049 [1] (see "Identifier" column and "Requirement" column in Table A.1.2.4.2-1) and Grafana Loki (see "Support by open source" and "Related capability of open source" column). The legend of "Support by open source" is the following:

"Yes": fully support the functional requirements.

"No": not support the functional requirements.

"Partial": partial support the functional requirements.

Table A.1.2.4.2-1: Comparison of Log Aggregator functional requirements and Grafana Loki

Identifier	Requirement	Support by open source	Related capability of open source
LogAggregator.001	The Log Aggregator function shall support the capability to collect different types of logs from different entities like the VNF instances (including the VNF's applications), NFV-MANO or NFVI (compute, storage, network resources) determined by a filter (see note 1).	Yes	Loki collects logs from various sources through integrations with log shippers like Fluentd or Fluent Bit. It also supports filtering during log querying using Loki's query language, which can filter by log attributes and labels.
LogAggregator.002	The Log Aggregator function shall support the capability to pre-process the logs (see note 2).	No	It does not provide built-in pre-processing capabilities. Logs are stored as they are collected. Any format conversion or enrichment can be handled by upstream tools or log shippers before they are sent to Loki.
LogAggregator.003	The Log Aggregator function shall support the capability to aggregate the logs in a configurable manner (see note 3).	No	It does not perform advanced log aggregation. It is designed for efficient log storage and querying. Aggregation of logs is done during querying using Loki's query language, rather than during ingestion or storage.
LogAggregator.004	The Log Aggregator function shall support the capability to store historical log records (see note 4).	Yes	Stores historical logs effectively. It uses a scalable storage backend optimized for managing large volumes of log data over extended periods, allowing for long-term log retention and analysis.
LogAggregator.005	The Log Aggregator function shall support the capability to expose (filtered) logs to authorized consumers.	Yes	It integrates with Grafana® to provide visualization and querying of logs. Access control and authorization for viewing logs are managed through Grafana®, allowing users to access and interact with logs based on their permissions.
NOTE 1: As an example for the case of VNF/VNFC instances, the filter shall support filtering of VNF/VNFC instances by type of the VNF/VNFC, vendor, host, zone, VNF instance identifier, etc. Also, it shall be able to filter by log attributes metric/log type, severity level, etc. NOTE 2: One form of pre-processing is to harmonize the format of the logs. NOTE 3: Examples of configurable forms of aggregation are to aggregate all logs based on criteria of log level, different instances belonging to the same VNF, VNF instances managed by the same VNFM, etc. NOTE 4: A use case to store historical log records is about using such records for further root-cause analysis.			

A.1.2.5 OpenSearch

A.1.2.5.1 Overview

This clause analyses the comparison of Log Aggregator functional requirements defined in clause 5.5 of ETSI GS NFV IFA 049 [1], and the open-source solution OpenSearch.

OpenSearch is an open-source search and analytics engine for scalable log and event data analysis. It offers powerful search, aggregation, and visualization tools for log management and real-time analytics.

A.1.2.5.2 Comparison

This clause shows comparison of Log Aggregator functional requirements defined in clause 5.5 of ETSI GS NFV-IFA 049 [1] (see "Identifier" column and "Requirement" column in Table A.1.2.5.2-1) and OpenSearch (see "Support by open source" and "Related capability of open source" column). The legend of "Support by open source" is the following:

"Yes": fully support the functional requirements.

"No": not support the functional requirements.

"Partial": partial support the functional requirements.

Table A.1.2.5.2-1: Comparison of Log Aggregator functional requirements and OpenSearch

Identifier	Requirement	Support by open source	Related capability of open source
LogAggregator.001	The Log Aggregator function shall support the capability to collect different types of logs from different entities like the VNF instances (including the VNF's applications), NFV-MANO or NFVI (compute, storage, network resources) determined by a filter (see note 1).	Yes	OpenSearch, via Logstash or Beats, can collect logs from various sources, including VMs, applications, containers, and infrastructure. Filters can be applied for log types, severity, etc.
LogAggregator.002	The Log Aggregator function shall support the capability to pre-process the logs (see note 2).	Partial	OpenSearch integrates with Logstash and Beats, which allow pre-processing of logs. This includes formatting, filtering, and enriching logs before they are stored in OpenSearch.
LogAggregator.003	The Log Aggregator function shall support the capability to aggregate the logs in a configurable manner (see note 3).	Yes	Supports configurable aggregation through Logstash pipelines or query-time aggregation. You can define log aggregation rules based on log level, source, application, etc
LogAggregator.004	The Log Aggregator function shall support the capability to store historical log records (see note 4).	Yes	It can store logs for long-term retention, allowing users to query historical log data for root cause analysis. Time-based indices help manage large volumes of log records efficiently.
LogAggregator.005	The Log Aggregator function shall support the capability to expose (filtered) logs to authorized consumers.	Yes	Offers RBAC to expose filtered logs to authorized users. Logs can be filtered and visualized using OpenSearch Dashboards.
NOTE 1: As an example for the case of VNF/VNFC instances, the filter shall support filtering of VNF/VNFC instances by type of the VNF/VNFC, vendor, host, zone, VNF instance identifier, etc. Also, it shall be able to filter by log attributes metric/log type, severity level, etc. NOTE 2: One form of pre-processing is to harmonize the format of the logs. NOTE 3: Examples of configurable forms of aggregation are to aggregate all logs based on criteria of log level, different instances belonging to the same VNF, VNF instances managed by the same VNFM, etc. NOTE 4: A use case to store historical log records is about using such records for further root-cause analysis.			

A.1.3 Comparison of Log Analyser functional requirements with relevant open-source solutions capabilities

A.1.3.1 ElastAlert 2

A.1.3.1.1 Overview

This clause analyses comparison of Log Analyser functional requirements defined in clause 5.6 of ETSI GS NFV IFA 049 [1], and open-source solution ElastAlert 2. It is an open-source alerting tool built on top of Elasticsearch, designed to monitor data and trigger alerts based on customizable rules. It enables users to define alert conditions and integrates with various notification services, making it for detecting anomalies and responding to real-time issues within Elasticsearch data.

A.1.3.1.2 Comparison

This clause shows comparison of Log Analyser functional requirements defined in clause 5.6 of ETSI GS NFV IFA 049 [1] as "Identifier" column and "Requirement" column in Table A.1.3.1.2.1 and ElastAlert 2 as "Support by open source" and "Related capability of open source" column. The legend of "Support by open source" is the following:

"Yes": fully support the functional requirements.

"No": not support the functional requirements.

"Partial": partial support the functional requirements.

Table A.1.3.1.2-1: Comparison of Log Analyser functional requirements and ElastAlert 2

Identifier	Requirement	Support by open source	Related capability of open source
LogAnalyser.001	The Log Analyser function shall support to analyse and process different types of logs based on a set of analysis functions (see note 1).	Yes	Supports functions like abnormal behaviour detection (including spikes, flatlines, backlist, whitelist), threshold crossing, and basic statistical processing.
LogAnalyser.002	The Log Analyser function shall support configuration of the analytics/processing to be applied (see note 2).	Yes	Configurable via YAML files. Allows setting thresholds, defining time windows, combining rules, and customizing analysis workflows.
LogAnalyser.003	The Log Analyser function shall support the capability to send notifications based on findings from the analysis of the logs.	Yes	Supports sending notifications via email, other relevant platforms, and custom scripts based on alerting conditions.
LogAnalyser.004	The Log Analyser function shall support the capability to expose analytics results to authorized consumers.	Yes	Can send alerts to external systems which could expose results, although, ElastAlert 2 doesn't natively provide an API or dashboard for analytics, but with custom integrations it can expose analytics results to the authorized consumers.
NOTE 1: Examples of analysis functions are abnormal behaviour detection, threshold cross, statistical processing, correlation of logs, etc. NOTE 2: Examples of configuration forms of the analytics are set threshold, define the composition of the analytic function from a set of basic analytic functions, etc.			

A.1.3.2 Coroot

A.1.3.2.1 Overview

This clause analyses comparison of Log Analyser functional requirements defined in clause 5.6 of ETSI GS NFV IFA 049 [1], and open-source solution Coroot.

Coroot is an open-source tool, which leverages eBPF to gather comprehensive telemetry data, including metrics, logs, and traces, offering visibility into system performance. It generates a detailed Service Map that visualizes the relationships and dependencies between services, helping users monitor and understand the health of their cloud-native infrastructure effectively.

NOTE: Currently, Coroot is included in the CNCF Landscape but is not listed among the CNCF Graduated or Incubating projects.

A.1.3.2.2 Comparison

This clause shows comparison of Log Analyser functional requirements defined in clause 5.6 of ETSI GS NFV IFA 049 [1] as "Identifier" column and "Requirement" column in Table A.1.3.2.2-1 and Coroot as "Support by open source" and "Related capability of open source" column. The legend of "Support by open source" is the following:

"Yes": fully support the functional requirements.

"No": not support the functional requirements.

"Partial": partial support the functional requirements.

Table A.1.3.2.2-1: Comparison of Log Analyser functional requirements and Coroot

Identifier	Requirement	Support by open source	Related capability of open source
LogAnalyser.001	The Log Analyser function shall support to analyse and process different types of logs based on a set of analysis functions (see note 1).	Yes	Supports detecting abnormal behaviour, such as performance issues and resource bottlenecks in applications.
LogAnalyser.002	The Log Analyser function shall support configuration of the analytics/processing to be applied (see note 2).	Yes	Provides options to configure alerting thresholds and customize log monitoring settings to track key performance metrics.
LogAnalyser.003	The Log Analyser function shall support the capability to send notifications based on findings from the analysis of the logs.	Yes	Integrates with various relevant platforms to notify users of detected issues or alerts based on analysis findings.
LogAnalyser.004	The Log Analyser function shall support the capability to expose analytics results to authorized consumers.	Yes	Provides dashboards and visual reports for easy access and sharing of analytics data with authorized users.
NOTE 1: Examples of analysis functions are abnormal behaviour detection, threshold cross, statistical processing, correlation of logs, etc. NOTE 2: Examples of configuration forms of the analytics are set threshold, define the composition of the analytic function from a set of basic analytic functions, etc.			

A.1.3.3 Grafana®

A.1.3.3.1 Overview

This clause analyses the comparison of Log Analyser functional requirements defined in clause 5.6 of ETSI GS NFV IFA 049 [1], and the open-source solution Grafana®.

Grafana® is an open-source observability platform for visualizing and analysing data from multiple sources. It is known for its customizable dashboards, alerting features, and real-time monitoring capabilities.

A.1.3.3.2 Comparison

This clause shows comparison of Log Analyser functional requirements defined in clause 5.6 of ETSI GS NFV-IFA 049 [1] as "Identifier" column and "Requirement" column in Table A.1.3.3.2-1 and Grafana® as "Support by open source" and "Related capability of open source" column. The legend of "Support by open source" is the following:

"Yes": fully support the functional requirements.

"No": not support the functional requirements.

"Partial": partial support the functional requirements.

Table A.1.3.3.2-1: Comparison of Log Analyser functional requirements and Grafana®

Identifier	Requirement	Support by open source	Related capability of open source
LogAnalyser.001	The Log Analyser function shall support to analyse and process different types of logs based on a set of analysis functions (see note 1).	Yes	Grafana® can integrate with Loki (for logs), supporting analysis such as threshold crossing, abnormal behaviour detection, and visual correlation of logs.
LogAnalyser.002	The Log Analyser function shall support configuration of the analytics/processing to be applied (see note 2).	Yes	Grafana® allows users to configure dashboards with thresholds, queries, and alerts. However, the core analytics are driven by external log processing tools like Loki, not Grafana® itself.
LogAnalyser.003	The Log Analyser function shall support the capability to send notifications based on findings from the analysis of the logs.	Yes	Supports alerts and notifications via integrations with other relevant platforms. These notifications are triggered by custom conditions applied to data (e.g. log errors, thresholds).
LogAnalyser.004	The Log Analyser function shall support the capability to expose analytics results to authorized consumers.	Yes	Provides RBAC to share dashboards and analytics results securely with authorized users through its web-based interface.
NOTE 1: Examples of analysis functions are abnormal behaviour detection, threshold cross, statistical processing, correlation of logs, etc. NOTE 2: Examples of configuration forms of the analytics are set threshold, define the composition of the analytic function from a set of basic analytic functions, etc.			

A.1.3.4 OpenSearch

A.1.3.4.1 Overview

This clause analyses the comparison of Log Analyser functional requirements defined in clause 5.6 of ETSI GS NFV IFA 049 [1], and the open-source solution OpenSearch.

OpenSearch is an open-source search and analytics engine for scalable log and event data analysis. It offers powerful search, aggregation, and visualization tools for log management and real-time analytics.

A.1.3.4.2 Comparison

This clause shows comparison of Log Analyser functional requirements defined in clause 5.6 of ETSI GS NFV-IFA 049 [1] as "Identifier" column and "Requirement" column in Table A.1.3.4.2-1 and OpenSearch as "Support by open source" and "Related capability of open source" column. The legend of "Support by open source" is the following:

"Yes": fully support the functional requirements.

"No": not support the functional requirements.

"Partial": partial support the functional requirements.

Table A.1.3.4.2-1: Comparison of Log Analyser functional requirements and OpenSearch

Identifier	Requirement	Support by open source	Related capability of open source
LogAnalyser.001	The Log Analyser function shall support to analyse and process different types of logs based on a set of analysis functions (see note 1).	Yes	OpenSearch supports log analysis functions such as anomaly detection, threshold-based alerting, statistical processing, and correlation of logs.
LogAnalyser.002	The Log Analyser function shall support configuration of the analytics/processing to be applied (see note 2).	Yes	Allows configuration through its alerting features, with customizable thresholds, time windows, and rule-based processing for log data.
LogAnalyser.003	The Log Analyser function shall support the capability to send notifications based on findings from the analysis of the logs.	Yes	Supports sending notifications through its alerting framework, it integrates with email, webhooks, and other relevant platforms for notifications based on log analysis.
LogAnalyser.004	The Log Analyser function shall support the capability to expose analytics results to authorized consumers.	Yes	OpenSearch Dashboards provide visual analytics results, which can be shared securely with authorized consumers via roles and access controls.
NOTE 1: Examples of analysis functions are abnormal behaviour detection, threshold cross, statistical processing, correlation of logs, etc. NOTE 2: Examples of configuration forms of the analytics are set threshold, define the composition of the analytic function from a set of basic analytic functions, etc.			

A.1.4 Comparison of Traffic Enforcer functional requirements with relevant open-source solutions capabilities

A.1.4.1 Cilium®

A.1.4.1.1 Overview

This clause analyses comparison of Traffic Enforcer functional requirements defined in clause 5.2 of ETSI GS NFV IFA 049 [1], and open-source solution Cilium®.

Cilium® is an open-source solution that provides networking, observability, and security via an eBPF based dataplane. Cilium® includes features like BGP, service mesh, and cross-cluster connectivity beyond basic Layer 3 networking for containers.

A.1.4.1.2 Comparison

This clause shows comparison of Traffic Enforcer functional requirements defined in clause 5.2 of ETSI GS NFV IFA 049 [1] as "Identifier" column and "Requirement" column in Table A.1.4.1.2-1 and Cilium® as "Support by open source" and "Related capability of open source" column. The legend of "Support by open source" is the following:

"Yes": fully support the functional requirements.

"No": not support the functional requirements.

"Partial": partial support the functional requirements.

Table A.1.4.1.2-1: Comparison of Traffic Enforcer functional requirements and Cilium®

Identifier	Requirement	Support by open source	Related capability of open source
TrafficEnforcer.001	The Traffic Enforcer function shall support the capability to perform traffic isolation and traffic rerouting of one or more VNFC instances (see note).	Yes	Cilium® uses eBPF-based policies for network isolation, restricting or blocking traffic across microservices. It also supports traffic rerouting through service mesh integration and applies rate-limiting measures for both partial and complete service isolation.
NOTE: Traffic isolation can be partial or full (i.e. lowering the traffic sent to a VNFC instance) or full (i.e. blocking the traffic sent to a VNFC instance).			

A.1.4.2 Istio®

A.1.4.2.1 Overview

This clause analyses comparison of Traffic Enforcer functional requirements defined in clause 5.2 of ETSI GS NFV IFA 049 [1], and open-source solution Istio®.

Istio® is an open-source service mesh that provides a way to manage microservices, facilitating service-to-service communication and offering features like traffic management, security, and observability. It allows developers to implement policies and telemetry for services without altering the application code, enabling better control over microservices architecture.

A.1.4.2.2 Comparison

This clause shows comparison of Traffic Enforcer functional requirements defined in clause 5.2 of ETSI GS NFV IFA 049 [1] as "Identifier" column and "Requirement" column in Table A.1.4.2.2-1 and Istio® as "Support by open source" and "Related capability of open source" column. The legend of "Support by open source" is the following:

"Yes": fully support the functional requirements.

"No": not support the functional requirements.

"Partial": partial support the functional requirements.

Table A.1.4.2.2-1: Comparison of Traffic Enforcer functional requirements and Istio®

Identifier	Requirement	Support by open source	Related capability of open source
TrafficEnforcer.001	The Traffic Enforcer function shall support the capability to perform traffic isolation and traffic rerouting of one or more VNFC instances (see note).	Yes	Istio® allows traffic isolation by applying policies to control traffic across microservices, either partially or fully. It supports traffic rerouting using service mesh features like virtual services and destination rules, enabling dynamic routing decisions and rerouting around affected VNFC instances.
NOTE: Traffic isolation can be partial or full (i.e. lowering the traffic sent to a VNFC instance) or full (i.e. blocking the traffic sent to a VNFC instance).			

A.1.4.3 Linkerd

A.1.4.3.1 Overview

This clause analyses comparison of Traffic Enforcer functional requirements defined in clause 5.2 of ETSI GS NFV IFA 049 [1], and open-source solution Linkerd.

Linkerd is an open-source service mesh that enhances microservices communication by providing features such as observability, load balancing, and security. It enables service-to-service interactions without requiring changes to application code, focusing on resilience and efficient resource use.

A.1.4.3.2 Comparison

This clause shows comparison of Traffic Enforcer functional requirements defined in clause 5.2 of ETSI GS NFV IFA 049 [1] as "Identifier" column and "Requirement" column in Table A.1.4.3.2-1 and Linkerd as "Support by open source" and "Related capability of open source" column. The legend of "Support by open source" is the following:

"Yes": fully support the functional requirements.

"No": not support the functional requirements.

"Partial": partial support the functional requirements.

Table A.1.4.3.2-1: Comparison of Traffic Enforcer functional requirements and Linkerd

Identifier	Requirement	Support by open source	Related capability of open source
TrafficEnforcer.001	The Traffic Enforcer function shall support the capability to perform traffic isolation and traffic rerouting of one or more VNFC instances (see note).	Yes	Linkerd enforces traffic policies using rate limiting and circuit breaking, enabling partial traffic isolation by controlling flow across pods, containers, or services. It also facilitates traffic rerouting through load balancing and failover features, allowing dynamic adjustments during service disruptions.
NOTE: Traffic isolation can be partial or full (i.e. lowering the traffic sent to a VNFC instance) or full (i.e. blocking the traffic sent to a VNFC instance).			

A.1.4.4 Envoy

A.1.4.4.1 Overview

This clause analyses comparison of Traffic Enforcer functional requirements defined in clause 5.2 of ETSI GS NFV IFA 049 [1], and open-source solution Envoy.

Envoy is an open-source edge and service proxy tailored for cloud-native applications. It manages service discovery, load balancing, and traffic routing while providing observability features for microservices communication. Operating at the application layer (Layer 7), Envoy simplifies complex service interactions without major architectural changes.

A.1.4.4.2 Comparison

This clause shows comparison of Traffic Enforcer functional requirements defined in clause 5.2 of ETSI GS NFV IFA 049 [1] as "Identifier" column and "Requirement" column in Table A.1.4.4.2-1 and Envoy as "Support by open source" and "Related capability of open source" column. The legend of "Support by open source" is the following:

"Yes": fully support the functional requirements.

"No": not support the functional requirements.

"Partial": partial support the functional requirements.

Table A.1.4.4.2-1: Comparison of Traffic Enforcer functional requirements and Envoy

Identifier	Requirement	Support by open source	Related capability of open source
TrafficEnforcer.001	The Traffic Enforcer function shall support the capability to perform traffic isolation and traffic rerouting of one or more VNFC instances (see note).	Yes	Envoy provides capabilities for traffic isolation and rerouting through its routing and service mesh features. It supports partial isolation by implementing rate limiting, which can control traffic across microservices. Additionally, Envoy facilitates traffic rerouting through load balancing and failure handling mechanisms.
NOTE: Traffic isolation can be partial or full (i.e. lowering the traffic sent to a VNFC instance) or full (i.e. blocking the traffic sent to a VNFC instance).			

A.1.5 Comparison of PaaS Service Policy Agent functional requirements with relevant open-source solutions capabilities

A.1.5.1 Open Policy Agent (OPA)

A.1.5.1.1 Overview

This clause analyses comparison of Policy Agent functional requirements specified in clause 5.11 of ETSI GS NFV IFA 049 [1], and open source solution Open Policy Agent (OPA).

OPA is an open source CNCF® project that includes a general purpose policy engine that decouples policy from application logic and separates policy decision from enforcement. It aims at improving the security and regulatory compliance of applications and underlying infrastructure in cloud native environments.

A.1.5.1.2 Comparison

This clause shows comparison of Policy Agent functional requirements defined in clause 5.11 of ETSI GS NFV IFA 049 [1] as "Identifier" column and "Requirement" column in Table A.1.5.1.2-1 and OPA as "Support by open source" and "Related capability of open source" column. The legend of "Support by open source" is the following:

"Yes": fully support the functional requirements.

"No": not support the functional requirements.

"Partial": partial support the functional requirements.

Table A.1.5.1.2-1: Comparison of Policy Agent functional requirements and OPA

Identifier	Requirement	Support by open source	Related capability of open source
PolicyAgent.001	The Policy Agent function shall support the capability to support automated decision making to ease administrative tasks (see note 1).	Yes	It is a policy decision point that can interact with various policy enforcement functions (e.g. Envoy Proxy, k8s API server) and can be used to prevent accidental operations such as e.g. deletion of namespaces.
PolicyAgent.002	The Policy Agent function shall support the capability to notify the consumers (e.g. OSS/BSS) about events related to policy managements actions (see note 2).	Partial	It can periodically report decision logs events using the Decision Log Service API. Security-sensitive attributes can be masked from the notifications.
PolicyAgent.003	The Policy Agent function shall support the capability to parse and execute VNF and VNF generic OAM functions policies (see note 3).	Partial	It requires to adopt Rego which is a flexible and expressive declarative language for writing policies.
PolicyAgent.004	The Policy Agent function shall support the capability to perform CRUD operations for policies upon request from a consumer (see note 4).	Yes	It provides REST API services to e.g. create, update, delete policies that can be stored with the data in a bundle registry.
NOTE 1: The Policy Agent function can interact with other VNF generic OAM functions to perform automated decision making with or without interaction with NFV-MANO components. See use case description in clause 4.4.2.6 of ETSI GR NFV-EVE 019 [i.2]. NOTE 2: Notifications sent by the Policy Agent are forwarded to the notifications subscriber (e.g. OSS/BSS) through the Notification Manager. See clause 4.2.1.5 of ETSI GS NFV-IFA 049 [1]. NOTE 3: Refer to description in clause 4.2.1.6 of ETSI GS NFV-IFA 049 [1] regarding the enforcement of policies. NOTE 4: The Policy Agent upon a request to create, delete or update a policy, updates accordingly a repository where the policies are stored. The policies can target VNF generic OAM functions and other PaaS Services and VNF/VNFC instances.			

A.1.6 Comparison of VNF Metrics Aggregator functional requirements with relevant open source solutions capabilities

A.1.6.1 Prometheus

A.1.6.1.1 Overview

This clause analyses comparison of VNF Metrics Aggregator functional requirements specified in clause 5.7 of ETSI GS NFV IFA 049 [1], and open source solution Prometheus.

Prometheus® is an open source CNCF® project that is designed to provide insights into system performance and health and is commonly used in cloud native environments to collect, store and query time series metric data.

A.1.6.1.2 Comparison

This clause shows comparison of VNF Metrics Aggregator functional requirements defined in clause 5.7 of ETSI GS NFV IFA 049 [1] as "Identifier" column and "Requirement" column in Table A.1.6.1.2-1 and Prometheus® as "Support by open source" and "Related capability of open source" column. The legend of "Support by open source" is the following:

"Yes": fully support the functional requirements.

"No": not support the functional requirements.

"Partial": partial support the functional requirements.

Table A.1.6.1.2-1: Comparison of VNF Metrics Aggregator functional requirements and Prometheus®

Identifier	Requirement	Support by open source	Related capability of open source
VNFMetricAggregator.001	The VNF Metrics Aggregator function shall support the capability to collect different types of metrics from entities determined by a filter (see notes 1 and 5).	Yes	It supports discovery mechanism of Prometheus® target endpoints that expose metrics for collection and supports filtering using PromQL.
VNFMetricAggregator.002	The VNF Metrics Aggregator function shall support the capability to pre-process the metrics (see note 2).	Yes	It includes ecosystem of community-maintained exporters that gather common metrics (e.g. hardware, OS metrics) and convert them into Prometheus-formatted metrics.
VNFMetricAggregator.003	The VNF Metrics Aggregator function shall support the capability to aggregate the metrics in a configurable manner (see note 3).	Yes	PromQL supports various built-in aggregation functions/operators that can be leveraged to aggregate and analyse time series metrics.
VNFMetricAggregator.004	The VNF Metrics Aggregator function shall support the capability to store time series metrics for records (see note 4).	Yes	It provides own time series backend for short to medium-term storage or can be integrated with Prometheus® compatible backends.
VNFMetricAggregator.005	The VNF Metrics Aggregator function shall support the capability to expose (filtered) metrics to authorized consumers.	Yes	It can be configured to integrate with various dashboards to visualize Prometheus-formatted metrics.
NOTE 1: The filter shall support operations like filtering of VNF/VNFC instances by type of the VNF/VNFC, vendor, host, zone, VNF instance identifier, etc. Also it shall be able to filter by metric/log type, severity level, etc. NOTE 2: One form of pre-processing is to harmonize the format of the metrics. NOTE 3: Examples of configurable forms of aggregation are to aggregate all metrics related to performance, aggregate metrics from different instances belonging to the same VNF, aggregate metrics of VNF instances managed by the same VNFM, etc. NOTE 4: Use cases for storing time services of metrics are for instance using the stored metrics for further root-cause analysis, abnormal behaviour detection, etc. NOTE 5: VNF Metrics include virtualisation-dependent as well as virtualisation-independent metrics like VNF's application metrics, VNF's connectivity and VNF's network performance metrics, NFVI related metrics, etc.			

A.1.6.2 OpenTelemetry Collector

A.1.6.2.1 Overview

This clause analyses comparison of VNF Metrics Aggregator functional requirements specified in clause 5.7 of ETSI GS NFV IFA 049 [1], and open source solution OpenTelemetry Collector.

OpenTelemetry Collector is an open source CNCF® project that is designed to receive, process and export telemetry data (logs, metrics and traces) to various backends in cloud native environments.

A.1.6.2.2 Comparison

This clause shows comparison of VNF Metrics Aggregator functional requirements defined in clause 5.7 of ETSI GS NFV IFA 049 [1] as "Identifier" column and "Requirement" column in Table A.1.6.2.2-1 and OpenTelemetry Collector as "Support by open source" and "Related capability of open source" column. The legend of "Support by open source" is the following:

"Yes": fully support the functional requirements.

"No": not support the functional requirements.

"Partial": partial support the functional requirements.

Table A.1.6.2.2-1: Comparison of VNF Metrics Aggregator functional requirements and OpenTelemetry Collector

Identifier	Requirement	Support by open source	Related capability of open source
VNFMetricAggregator.001	The VNF Metrics Aggregator function shall support the capability to collect different types of metrics from entities determined by a filter (see notes 1 and 5).	Yes	It supports receiving metrics data from multiple sources by leveraging community-maintained receivers (e.g. Prometheus® receiver, Host Metrics receiver) or by integrating custom-built receivers using the OpenTelemetry Collector Builder.
VNFMetricAggregator.002	The VNF Metrics Aggregator function shall support the capability to pre-process the metrics (see note 2).	Yes	One or more pipelines of processors can be leveraged to pre-process the metrics and perform operations such as format conversion, enrichment and harmonization before forwarding the processed data to exporters
VNFMetricAggregator.003	The VNF Metrics Aggregator function shall support the capability to aggregate the metrics in a configurable manner (see note 3).	Yes	Community maintained processors (e.g. Transform processor) can be configured to aggregate and transform metrics using the OTTL. Custom-based processors can also be integrated and configured allowing for customized metrics aggregation.
VNFMetricAggregator.004	The VNF Metrics Aggregator function shall support the capability to store time series metrics for records (see note 4).	Yes	OTLP exporters can be configured to export metrics to various backends that support OTLP format.
VNFMetricAggregator.005	The VNF Metrics Aggregator function shall support the capability to expose (filtered) metrics to authorized consumers.	Yes	OTLP exporters can be configured to integrate with various dashboards to visualize OTLP-formatted metrics.

NOTE 1: The filter shall support operations like filtering of VNF/VNFC instances by type of the VNF/VNFC, vendor, host, zone, VNF instance identifier, etc. Also it shall be able to filter by metric/log type, severity level, etc.
NOTE 2: One form of pre-processing is to harmonize the format of the metrics.
NOTE 3: Examples of configurable forms of aggregation are to aggregate all metrics related to performance, aggregate metrics from different instances belonging to the same VNF, aggregate metrics of VNF instances managed by the same VNFM, etc.
NOTE 4: Use cases for storing time services of metrics are for instance using the stored metrics for further root-cause analysis, abnormal behaviour detection, etc.
NOTE 5: VNF Metrics include virtualisation-dependent as well as virtualisation-independent metrics like VNF's application metrics, VNF's connectivity and VNF's network performance metrics, NFVI related metrics, etc.

A.1.6.3 VictoriaMetrics

A.1.6.3.1 Overview

This clause analyses comparison of VNF Metrics Aggregator functional requirements specified in clause 5.7 of ETSI GS NFV IFA 049 [1], and open source solution VictoriaMetrics.

VictoriaMetrics is an open-source project that extends the Prometheus® capabilities. It is a time series database that enables full-stack metric collection, rule evaluation, and alert routing with deduplication, grouping, and silencing features.

A.1.6.3.2 Comparison

This clause shows comparison of VNF Metrics Aggregator functional requirements defined in clause 5.7 of ETSI GS NFV IFA 049 [1] as "Identifier" column and "Requirement" column in Table A.1.6.3.2-1 and VictoriaMetrics as "Support by open source" and "Related capability of open source" column. The legend of "Support by open source" is the following:

"Yes": fully support the functional requirements.

"No": not support the functional requirements.

"Partial": partial support the functional requirements.

Table A.1.6.3.2-1: Comparison of VNF Metrics Aggregator functional requirements and VictoriaMetrics

Identifier	Requirement	Support by open source	Related capability of open source
VNFMetricAggregator.001	The VNF Metrics Aggregator function shall support the capability to collect different types of metrics from entities determined by a filter (see notes 1 and 5).	Yes	Provides full metric ingestion control through VMAgent custom resource, including label-based filtering for selective metric collection, automatic target discovery, and dynamic metric relabelling for data consistency.
VNFMetricAggregator.002	The VNF Metrics Aggregator function shall support the capability to pre-process the metrics (see note 2).	Yes	Supports real-time metric processing including label and value modifications, and conditional metric filtering during ingestion.
VNFMetricAggregator.003	The VNF Metrics Aggregator function shall support the capability to aggregate the metrics in a configurable manner (see note 3).	Yes	Supports run-time metric aggregation using PromQL functions (sum(), avg(), etc.). It also provides VMAlert custom resource for pre-calculated metrics via recording rules and custom alerting logic.
VNFMetricAggregator.004	The VNF Metrics Aggregator function shall support the capability to store time series metrics for records (see note 4).	Yes	Supports optimized long-term storage with high compression ratios. It also supports configurable retention policies and cost-efficient backups to cloud storage.
VNFMetricAggregator.005	The VNF Metrics Aggregator function shall support the capability to expose (filtered) metrics to authorized consumers.	Partial	Provides HTTP Query API for metrics and alert status exposure. It supports TLS encryption and basic auth, while advanced RBAC requires reverse proxies or API gateways.

NOTE 1: The filter shall support operations like filtering of VNF/VNFC instances by type of the VNF/VNFC, vendor, host, zone, VNF instance identifier, etc. Also it shall be able to filter by metric/log type, severity level, etc.
NOTE 2: One form of pre-processing is to harmonize the format of the metrics.
NOTE 3: Examples of configurable forms of aggregation are to aggregate all metrics related to performance, aggregate metrics from different instances belonging to the same VNF, aggregate metrics of VNF instances managed by the same VNFM, etc.
NOTE 4: Use cases for storing time services of metrics are for instance using the stored metrics for further root-cause analysis, abnormal behaviour detection, etc.
NOTE 5: VNF Metrics include virtualisation-dependent as well as virtualisation-independent metrics like VNF's application metrics, VNF's connectivity and VNF's network performance metrics, NFVI related metrics, etc.

A.1.7 Comparison of VNF Metrics Analyser functional requirements with relevant open source solutions capabilities

A.1.7.1 Coroot

A.1.7.1.1 Overview

This clause analyses comparison of VNF Metrics Analyser functional requirements specified in clause 5.8 of ETSI GS NFV IFA 049 [1], and open source solution Coroot.

Coroot is an open-source project, which is powered by eBPF and can natively be integrated with Kubernetes® to offer visibility into system performance and application health in cloud native environments.

A.1.7.1.2 Comparison

This clause shows comparison of VNF Metrics Analyser functional requirements defined in clause 5.8 of ETSI GS NFV IFA 049 [1] as "Identifier" column and "Requirement" column in Table A.1.7.1.2-1 and Coroot as "Support by open source" and "Related capability of open source" column. The legend of "Support by open source" is the following:

"Yes": fully support the functional requirements.

"No": not support the functional requirements.

"Partial": partial support the functional requirements.

Table A.1.7.1.2-1: Comparison of VNF Metrics Analyser functional requirements and Coroot

Identifier	Requirement	Support by open source	Related capability of open source
VNFMetricAnalyser.001	The VNF Metrics Analyser function shall support the capability to analyse and process different types of metrics based on a set of analysis functions (see note 1).	Yes	It can natively be integrated with Prometheus, supports detecting abnormal behaviour such as performance issues and resource bottlenecks in applications and can generate audit reports, e.g. about DNS related issues.
VNFMetricAnalyser.002	The VNF Metrics Analyser function shall support the capability to provide configuration of the analytics/processing of metrics to be applied (see note 2).	Partial	It provides options to configure/override inspection thresholds for common metrics such as node CPU utilization or to define custom SLOs to ensure that e.g. a certain percentage of requests are performed faster than a given latency threshold.
VNFMetricAnalyser.003	The VNF Metrics Analyser function shall support the capability to send notifications based on findings from the analysis of the metrics.	Partial	It supports sending messages about incidents and status of deployments via webhooks.
VNFMetricAnalyser.004	The VNF Metrics Aggregator function shall support the capability to expose analytics report to authorized consumers.	Yes	It provides dashboards and visual reports for easy access and sharing of analytics data with authorized users.
NOTE 1: Examples of analysis functions are abnormal behaviour detection, threshold crossing, statistical processing, etc. NOTE 2: Examples of configuration forms of the analytics are set thresholds, define the composition of the analytic function from a set of basic analytic functions.			

A.1.7.2 OpenSearch

A.1.7.2.1 Overview

This clause analyses the comparison of VNF Metrics Analyser functional requirements specified in clause 5.8 of ETSI GS NFV IFA 049 [1], and the open source solution OpenSearch.

OpenSearch is an open-source search and analytics engine for scalable metrics and event data analysis. It offers powerful search, aggregation, and visualization tools for real-time metrics monitoring and observability.

A.1.7.2.2 Comparison

This clause shows comparison of VNF Metrics Analyser functional requirements defined in clause 5.8 of ETSI GS NFV IFA 049 [1] as "Identifier" column and "Requirement" column in Table A.1.7.2.2-1 and OpenSearch as "Support by open source" and "Related capability of open source" column. The legend of "Support by open source" is the following:

"Yes": fully support the functional requirements.

"No": not support the functional requirements.

"Partial": partial support the functional requirements.

Table A.1.7.2.2-1: Comparison of VNF Metrics Analyser functional requirements and OpenSearch

Identifier	Requirement	Support by open source	Related capability of open source
VNFMetricAnalyser.001	The VNF Metrics Analyser function shall support the capability to analyse and process different types of metrics based on a set of analysis functions (see note 1).	Yes	Supports metrics analysis through custom queries, aggregations, and threshold-based alerting. It supports various statistical processing methods, including histograms, to help users analyse trends and patterns in their data. Additionally, the Anomaly Detection plugin enhances these capabilities by automatically identifying unusual behaviour and trends.
VNFMetricAnalyser.002	The VNF Metrics Analyser function shall support the capability to provide configuration of the analytics/processing of metrics to be applied (see note 2).	Yes	Supports configurable metric analytics through custom queries, dashboards, and alerting rules. Users can set thresholds, define aggregation logic, and create alerting conditions. Additionally, the Anomaly Detection plugin enables advanced configuration, such as setting custom detection intervals and fine-tuning anomaly detection models.
VNFMetricAnalyser.003	The VNF Metrics Analyser function shall support the capability to send notifications based on findings from the analysis of the metrics.	Yes	It includes built-in alerting capabilities that enable the creation of monitors and triggers to send notifications based on metric conditions. Notifications can be routed to various channels such as email, and webhooks.
VNFMetricAnalyser.004	The VNF Metrics Analyser function shall support the capability to expose the metrics analytics results to authorized consumers.	Yes	It provides RBAC and authentication mechanisms to restrict analytics access to authorized users. It enables secure metrics visualization in OpenSearch Dashboards, allowing users to apply permissions and share data while maintaining data security.
NOTE 1: Examples of analysis functions are abnormal behaviour detection, threshold crossing, statistical processing, etc. NOTE 2: Examples of configuration forms of the analytics are set thresholds, define the composition of the analytic function from a set of basic analytic functions.			

A.1.7.3 VictoriaMetrics

A.1.7.3.1 Overview

This clause analyses the comparison of VNF Metrics Analyser functional requirements specified in clause 5.8 of ETSI GS NFV IFA 049 [1], and the open source solution VictoriaMetrics.

VictoriaMetrics is an open-source project that extends the Prometheus® capabilities. It is a time series database that enables full-stack metric collection, rule evaluation, and alert routing with deduplication, grouping, and silencing features.

A.1.7.3.2 Comparison

This clause shows comparison of VNF Metrics Analyser functional requirements defined in clause 5.8 of ETSI GS NFV IFA 049 [1] as "Identifier" column and "Requirement" column in Table A.1.7.3.2-1 and VictoriaMetrics as "Support by open source" and "Related capability of open source" column. The legend of "Support by open source" is the following:

"Yes": fully support the functional requirements.

"No": not support the functional requirements.

"Partial": partial support the functional requirements.

Table A.1.7.3.2-1: Comparison of VNF Metrics Analyser functional requirements and VictoriaMetrics

Identifier	Requirement	Support by open source	Related capability of open source
VNFMetricAnalyser.001	The VNF Metrics Analyser function shall support the capability to analyse and process different types of metrics based on a set of analysis functions (see note 1).	Partial	Supports time-series analysis using MetricsQL, enabling threshold-based alerts, rate calculations, and statistical aggregations viaVMRule. However, advanced analytics such as anomaly detection or machine learning-based behaviour prediction require additional customization (e.g. via VMAnomaly).
VNFMetricAnalyser.002	The VNF Metrics Analyser function shall support the capability to provide configuration of the analytics/processing of metrics to be applied (see note 2).	Partial	Supports rule-based configuration through VMRule defined recording and alerting rules, providing the capability to define thresholds and compose analytic expressions using MetricsQL. However, lacks advanced pipeline management features or runtime configurability beyond static rule definitions.
VNFMetricAnalyser.003	The VNF Metrics Analyser function shall support the capability to send notifications based on findings from the analysis of the metrics.	Yes	Supports generating alerts based on the rules defined in VMRule resource(s), and forwards them to VictoriaMetrics Alertmanager instances. VictoriaMetrics Alertmanager takes care of routing, grouping, suppressing, and delivering notifications to external systems such as email, webhooks, or custom endpoints. It also supports message templating and the ability to silence alerts when needed.
VNFMetricAnalyser.004	The VNF Metrics Analyser function shall support the capability to expose the metrics analytics results to authorized consumers.	Partial	Provides Query API to expose analytical results including metrics from recording rules and firing alert statuses. While it supports TLS encryption and basic client certificate validation, implementing advanced authentication and authorization (e.g. RBAC) requires external tools such as reverse proxies or OAuth2 gateways.
NOTE 1: Examples of analysis functions are abnormal behaviour detection, threshold crossing, statistical processing, etc. NOTE 2: Examples of configuration forms of the analytics are set thresholds, define the composition of the analytic function from a set of basic analytic functions.			

A.1.8 Comparison of Notification Manager functional requirements with relevant open-source solutions capabilities

A.1.8.1 Prometheus Alertmanager

A.1.8.1.1 Overview

This clause analyses comparison of Notification manager functional requirements defined in clause 5.13 of ETSI GS NFV IFA 049 [1], and open-source solution Prometheus Alertmanager. It is an open-source alerting tool designed to monitor data and trigger alerts based on customizable rules. It enables users to define alert conditions and integrates with various notification services, making it for detecting anomalies and responding to real-time issues.

A.1.8.1.2 Comparison

This clause shows comparison of Notification Manager functional requirements defined in clause 5.13 of ETSI GS NFV IFA 049 [1] as "Identifier" column and "Requirement" column in Table A.1.8.1.2-1 and Prometheus alertmanager as "Support by open source" and "Related capability of open source" column. The legend of "Support by open source" is the following:

"Yes": fully support the functional requirements.

"No": not support the functional requirements.

"Partial": partial support the functional requirements.

Table A.1.8.1.2-1: Comparison of Notification Manager functional requirements and Prometheus Alertmanager

Identifier	Requirement	Support by open source	Related capability of open source
NotificationManager.001	The Notification Manager shall support the capability to receive notifications sent by other VNF generic OAM functions and other PaaS Services.	Yes	Prometheus Alertmanager receives alerts from Prometheus Server and can process external alerts.
NotificationManager.002	The Notification Manager shall support the capability to process (e.g. group, deduplicate) received notifications (see note 1).	Yes	Prometheus Alertmanager provides alert grouping and deduplication based on alert labels.
NotificationManager.003	The Notification Manager shall support the capability to route processed notification to authorized consumers.	Yes	Prometheus Alertmanager routes notifications to various channels, such as email, webhooks, and other custom endpoints.
NotificationManager.004	The Notifications manager shall support the capability for a consumer to manage subscriptions to notifications about events reported by VNF generic OAM functions and other PaaS Services (see notes 1 and 2)	Partial	Prometheus Alertmanager does not have a built-in subscription management mechanism but allows selective routing via label-based filtering. It requires external integration or API-based subscription handling.
NOTE 1: Notifications sent by VNF generic OAM functions and other PaaS Services to the Notification Manager, can be sent on demand or automatically to the Notification Manager. NOTE 2: Subscription management includes creation and termination of subscriptions to notifications.			

A.1.8.2 Argo®

A.1.8.2.1 Overview

This clause analyses comparison of Notification Manager functional requirements defined in clause 5.13 of ETSI GS NFV IFA 049 [1], and open-source solution Argo®.

Argo® is an open-source Kubernetes®-native event-driven workflow automation tool.

A.1.8.2.2 Comparison

This clause shows comparison of Notification Manger functional requirements defined in clause 5.13 of ETSI GS NFV IFA 049 [1] as "Identifier" column and "Requirement" column in Table A.1.8.2.2-1 and Argo® as "Support by open source" and "Related capability of open source" column. The legend of "Support by open source" is the following:

"Yes": fully support the functional requirements.

"No": not support the functional requirements.

"Partial": partial support the functional requirements.

Table A.1.8.2.2-1: Comparison of Notification Manager functional requirements and Argo®

Identifier	Requirement	Support by open source	Related capability of open source
NotificationManager.001	The Notification Manager shall support the capability to receive notifications sent by other VNF generic OAM functions and other PaaS Services.	Partial	Supports various sources, native VNF OAM integration might require custom adapters.
NotificationManager.002	The Notification Manager shall support the capability to process (e.g. group, deduplicate) received notifications (see note 1).	Partial	Allows event filtering and transformation before routing. It requires additional logic for deduplication.
NotificationManager.003	The Notification Manager shall support the capability to route processed notification to authorized consumers.	Yes	Supports routing and notification delivery based on defined triggers to Kubernetes® services, workflows, and other consumers.
NotificationManager.004	The Notifications manager shall support the capability for a consumer to manage subscriptions to notifications about events reported by VNF generic OAM functions and other PaaS Services (see notes 1 and 2).	Partial	Subscription handling exists lacks a dynamic self-service subscription model.
NOTE 1: Notifications sent by VNF generic OAM functions and other PaaS Services to the Notification Manager, can be sent on demand or automatically to the Notification Manager. NOTE 2: Subscription management includes creation and termination of subscriptions to notifications.			

A.1.8.3 Kafka

A.1.8.3.1 Overview

This clause analyses the comparison of Notification Manager functional requirements defined in clause 5.13 of ETSI GS NFV IFA 049 [1], and the open-source solution Kafka.

Kafka is an open-source distributed event streaming platform designed for high-throughput data distribution. It serves as a central hub for receiving, storing, and routing events to multiple consumers in real-time using a publish-subscribe messaging model, while stream processing is enabled through external frameworks like Kafka Streams.

A.1.8.3.2 Comparison

This clause shows comparison of Notification Managers functional requirements defined in clause 5.13 of ETSI GS NFV IFA 049 [1] as "Identifier" column and "Requirement" column in Table A.1.8.3.2-1 and Kafka as "Support by open source" and "Related capability of open source" column. The legend of "Support by open source" is the following:

"Yes": fully support the functional requirements.

"No": not support the functional requirements.

"Partial": partial support the functional requirements.

Table A.1.8.3.2-1: Comparison of Notification Manager functional requirements and Kafka

Identifier	Requirements	Support by open source	Related capability of open source
NotificationManager.001	The Notification Manager shall support the capability to receive notifications sent by other VNF generic OAM functions and other PaaS Services.	Yes	Provides a publish-subscribe model that allows event producers (VNFs, PaaS services, etc.) to send notifications to Kafka topics.
NotificationManager.002	The Notification Manager shall support the capability to process (e.g. group, deduplicate) received notifications (see note 1).	Partial	Does not process notifications itself, but Kafka Streams, or deduplication and event processing can be implemented by external consumer.
NotificationManager.003	The Notification Manager shall support the capability to route processed notification to authorized consumers.	Yes	Provides topic-based routing and consumer group model which allow notifications to be routed dynamically to multiple subscribers.
NotificationManager.004	The Notifications manager shall support the capability for a consumer to manage subscriptions to notifications about events reported by VNF generic OAM functions and other PaaS Services (see notes 1 and 2).	Partial	Follows a pub-sub model, where consumers automatically subscribe to Kafka topics by joining a consumer group. Subscription management needs to be handled by external authentication mechanisms (e.g. Kafka ACLs, OAuth, or RBAC in applications).
NOTE 1: Notifications sent by VNF generic OAM functions and other PaaS Services to the Notification Manager, can be sent on demand or automatically to the Notification Manager. NOTE 2: Subscription management includes creation and termination of subscriptions to notifications.			

A.1.8.4 Sensu

A.1.8.4.1 Overview

This clause analyses the comparison of Notification Manager functional requirements defined in clause 5.13 of ETSI GS NFV IFA 049 [1], and the open-source solution Sensu.

Sensu is an open-source monitoring and observability tool designed for dynamic cloud-native environments, hybrid infrastructure, and traditional data centers. It provides event-driven monitoring, enabling real-time health checks, metric collection, and alerting across distributed systems.

A.1.8.4.2 Comparison

This clause shows comparison of Notification Managers functional requirements defined in clause 5.13 of ETSI GS NFV IFA 049 [1] as "Identifier" column and "Requirement" column in Table A.1.8.4.2-1 and Sensu as "Support by open source" and "Related capability of open source" column. The legend of "Support by open source" is the following:

"Yes": fully support the functional requirements.

"No": not support the functional requirements.

"Partial": partial support the functional requirements.

Table A.1.8.4.2-1: Comparison of Notification Manager functional requirements and Sensu

Identifier	Requirements	Support by open source	Related capability of open source
NotificationManager.001	The Notification Manager shall support the capability to receive notifications sent by other VNF generic OAM functions and other PaaS Services.	Yes	Enables event reception from various sources using event brokers and sources.
NotificationManager.002	The Notification Manager shall support the capability to process (e.g. group, deduplicate) received notifications (see note 1).	Partial	Provides event routing but does not natively handle deduplication or grouping like Prometheus Alertmanager. However, it can be extended via custom processing logic.
NotificationManager.003	The Notification Manager shall support the capability to route processed notification to authorized consumers.	Partial	Supports sending processed notifications to authorized consumers via handlers, RBAC, and subscriptions It forwards notifications but does not store them for later consumption.
NotificationManager.004	The Notifications manager shall support the capability for a consumer to manage subscriptions to notifications about events reported by VNF generic OAM functions and other PaaS Services (see notes 1 and 2).	Partial	Allows users to configure event routing and filtering, but it does not provide a built-in mechanism for dynamic subscription management, such as creating or terminating subscriptions automatically.
NOTE 1: Notifications sent by VNF generic OAM functions and other PaaS Services to the Notification Manager, can be sent on demand or automatically to the Notification Manager. NOTE 2: Subscription management includes creation and termination of subscriptions to notifications.			

A.1.9 Comparison of PaaS Service Configuration Server functional requirements with relevant open-source solutions capabilities

A.1.9.1 Schema-driven Configuration (SDCIO)

A.1.9.1.1 Overview

This clause analyses comparison of Configuration Server functional requirements specified in clause 5.14 of ETSI GS NFV IFA 049 [1], and open source solution Schema-driven Configuration (SDCIO in abbreviation).

SDCIO is an open source project that uses schema-driven approaches for configuring in a declarative way PNFs, VM-based and container-based VNFs.

A.1.9.1.2 Comparison

This clause shows comparison of Configuration Server functional requirements defined in clause 5.14 of ETSI GS NFV IFA 049 [1] as "Identifier" column and "Requirement" column in Table A.1.9.1.2-1 and SDCIO as "Support by open source" and "Related capability of open source" column. The legend of "Support by open source" is the following:

"Yes": fully support the functional requirements.

"No": not support the functional requirements.

"Partial": partial support the functional requirements.

Table A.1.9.1.2-1: Comparison of Configuration Server functional requirements and SDCIO

Identifier	Requirement	Support by open source	Related capability of open source
ConfigServer.001	The Configuration Server shall support the capability of storing configuration data.	Yes	SDCIO Config API Server can be used to store configuration artifacts and may be integrated with GitOps solutions. In the latter case, network configurations are stored in Git repositories.
ConfigServer.002	The Configuration Server shall support the capability of converting configuration data between different formats.	Yes	Vendor-specific data formats (e.g. Netconf/XML) can be transformed to a unified abstract format (e.g. JSON) so that desired configuration states can be applied via Intent CRs and stored in the SDCIO Config API Server backend.
ConfigServer.003	The Configuration Server should support the capability to validate configuration data based on data schemas (see note 1).	Partial	It has basic validation capabilities (e.g. dry run validation) but does not support semantic validation of configuration data schemas.
ConfigServer.004	The Configuration Server shall support the capability of version control configuration data.	Yes	It leverages Kubernetes® native version control capabilities for managing network configurations.
ConfigServer.005	The Configuration Server shall support the capability of providing the configuration data to consumers that request to fetch such configuration data.	Yes	It supports CRUD operations for network configurations.
ConfigServer.006	The Configuration Server shall support the capability keep information about the stored configuration data (see note 2).	Yes	It also supports Repository Management APIs and GitOps REST APIs for easy integration with GitOps solutions.
ConfigServer.007	The Configuration Server shall support the capability to notify about events and changes of configuration data.	Partial	It leverages de-facto Kubernetes® capabilities (i.e. events related to status updates on custom resources) rather than external alerting mechanisms. For example, in the case of configuration deviation, Config CR status is updated by the DeviationWatcher and drift can be checked e.g. using GitOps drift API.
NOTE 1: Data schemas can be onboarded onto the Configuration Server or provided during operations issued by a consumer. NOTE 2: Information about the stored configuration data includes, but it is not limited to format of the data, last modification date of the data, version of the stored data.			

A.2 Comparison of the VNF generic OAM functions interface requirements against the considered open-source solutions

A.2.1 Overview

This clause analyses comparison of VNF generic OAM functions interface requirements specified in ETSI GS NFV-IFA 049 [1] against the considered open-source solutions as specified in clause A.1.

A.2.2 Comparison of Log Aggregator Interface requirements with considered open-source solutions capabilities

A.2.2.1 Fluent Bit

This clause shows comparison of Log Aggregator interface requirements defined in clause 6.2.4 of ETSI GS NFV IFA 049 [1] (see "Identifier" column and "Requirement" column in Table A.2.2.1-1) and Fluent Bit (see "Support by open source" and "Related capability of open source" column). The legend of "Support by open source" is the following:

"Yes": fully support the interface requirements.

"No": not support the interface requirements.

"Partial": partial support the interface requirements.

Table A.2.2.1-1: Comparison of Log Aggregator interface requirements and Fluent Bit

Identifier	Requirement	Support by open source	Related capability of open source
VNFLogAggregator.001	The VNF Log Aggregator function shall support producing the Log Exposure Interface (see note).	Yes	Supports various output plugins, such as HTTP, Prometheus®, OpenTelemetry™, OpenSearch, allow Fluent Bit to expose logs to multiple destinations.
LogAggr.Expose.001	The Log Exposure Interface shall support exposing the logs to authorized consumers.	Yes	Supports various output plugins allow secure log transmission with TLS/SSL encryption and, in some cases, Basic Authentication.
LogAggr.Expose.002	The Log Exposure Interface shall support the capability to support filtering of the logs.	Yes	Supports various filter plugins such as Grep, Modify, Parser, Kubernetes® Metadata, can include/exclude logs based on patterns, restructure logs for easier analysis, and are configured via the Fluent Bit configuration file, facilitating customizable log exposure.
NOTE: Refer to the support of capabilities of log collection by the log aggregator function specified in clause 8.2 of ETSI GS NFV IFA 049 [1].			

A.2.2.2 Fluentd

This clause shows comparison of Log Aggregator interface requirements defined in clause 6.2.4 of ETSI GS NFV IFA 049 [1] see "Identifier" column and "Requirement" column in Table A.2.2.2-1) and Fluentd (see "Support by open source" and "Related capability of open source" column). The legend of "Support by open source" is the following:

"Yes": fully support the interface requirements.

"No": not support the interface requirements.

"Partial": partial support the interface requirements.

Table A.2.2.2-1: Comparison of Log Aggregator interface requirements and Fluentd

Identifier	Requirement	Support by open source	Related capability of open source
VNFLogAggregator.001	The VNF Log Aggregator function shall support producing the Log Exposure Interface (see note).	Yes	Supports various output plugins, such as out_http (HTTP), out_opensearch (OpenSearch), out_s3 (S3), and out_forward (for other Fluent Bit nodes), allow Fluentd to expose logs to multiple destinations.
LogAggr.Expose.001	The Log Exposure Interface shall support exposing the logs to authorized consumers.	Yes	Supports various output plugins allow secure log transmission with TLS/SSL encryption and, in some cases, Basic Authentication.
LogAggr.Expose.002	The Log Exposure Interface shall support the capability to support filtering of the logs.	Yes	Supports various filter plugins such as Grep, Parser, Record Transformer, allows for tailored log exposure based on specific criteria, improving log management and analysis capabilities.
NOTE: Refer to the support of capabilities of log collection by the log aggregator function specified in clause 8.2 of ETSI GS NFV IFA 049 [1].			

A.2.2.3 OpenTelemetry Collector

This clause shows comparison of Log Aggregator interface requirements defined in clause 6.2.4 of ETSI GS NFV IFA 049 [1] (see "Identifier" column and "Requirement" column in Table A.2.2.3-1) and OpenTelemetry Collector (see "Support by open source" and "Related capability of open source" column). The legend of "Support by open source" is the following:

"Yes": fully support the interface requirements.

"No": not support the interface requirements.

"Partial": partial support the interface requirements.

Table A.2.2.3-1: Comparison of Log Aggregator interface requirements and OpenTelemetry Collector

Identifier	Requirement	Support by open source	Related capability of open source
VNFLogAggregator.001	The VNF Log Aggregator function shall support producing the Log Exposure Interface (see note).	Yes	Supports various receivers such as OTLP, HTTP, file, Prometheus to ingest logs from multiple sources, allowing for effective log aggregation.
LogAggr.Expose.001	The Log Exposure Interface shall support exposing the logs to authorized consumers.	Yes	Supports TLS/SSL for secure transport and authentication options for HTTP receivers, to secure log transmission options and can enforce access control through authentication mechanisms to protect log data.
LogAggr.Expose.002	The Log Exposure Interface shall support the capability to support filtering of the logs.	Yes	The Processor supports filtering and modifying capabilities, to filter and transform logs, allowing users to manage which logs are exposed based on specific criteria.
NOTE: Refer to the support of capabilities of log collection by the log aggregator function specified in clause 8.2 of ETSI GS NFV IFA 049 [1].			

A.2.2.4 Grafana Loki

This clause shows comparison of Log Aggregator interface requirements defined in clause 6.2.4 of ETSI GS NFV IFA 049 [1] (see "Identifier" column and "Requirement" column in Table A.2.2.4-1) and Grafana Loki (see "Support by open source" and "Related capability of open source" column). The legend of "Support by open source" is the following:

"Yes": fully support the interface requirements.

"No": not support the interface requirements.

"Partial": partial support the interface requirements.

Table A.2.2.4-1: Comparison of Log Aggregator interface requirements and Grafana Loki

Identifier	Requirement	Support by open source	Related capability of open source
VNFLogAggregator.001	The VNF Log Aggregator function shall support producing the Log Exposure Interface (see note).	Yes	Can ingest logs from multiple sources, including applications and system logs, through the Loki API (/loki/api/v1/push for log ingestion) and Promtail (collects and sends logs to loki), making it adaptable to various logging scenarios.
LogAggr.Expose.001	The Log Exposure Interface shall support exposing the logs to authorized consumers.	Yes	Integrates with Grafana® over TLS, for user management and access control, ensuring that logs can be securely accessed by authorized consumers. It also supports Basic Auth and Token-based authentication.
LogAggr.Expose.002	The Log Exposure Interface shall support the capability to support filtering of the logs.	Yes	Supports (LogQL, Loki's query language for log retrieval) for complex queries for log filtering and retrieval, allowing users to specify criteria for effective log management.
NOTE: Refer to the support of capabilities of log collection by the log aggregator function specified in clause 8.2 of ETSI GS NFV IFA 049 [1].			

A.2.2.5 OpenSearch

This clause shows comparison of Log Aggregator interface requirements defined in clause 6.2.4 of ETSI GS NFV IFA 049 [1] (see "Identifier" column and "Requirement" column in Table A.2.2.5-1) and OpenSearch (see "Support by open source" and "Related capability of open source" column). The legend of "Support by open source" is the following:

"Yes": fully support the interface requirements.

"No": not support the interface requirements.

"Partial": partial support the interface requirements.

Table A.2.2.5-1: Comparison of Log Aggregator interface requirements and OpenSearch

Identifier	Requirement	Support by open source	Related capability of open source
VNFLogAggregator.001	The VNF Log Aggregator function shall support producing the Log Exposure Interface (see note).	Yes	Supports ingest APIs such as Bulk API (/_bulk) for batch log ingestion and Index API, allowing logs to be sent directly to OpenSearch.
LogAggr.Expose.001	The Log Exposure Interface shall support exposing the logs to authorized consumers.	Yes	Supports robust security features such as Security APIs (/_security), Security plugin REST API, ensuring that only authorized users can access logs through authentication and role-based access control.
LogAggr.Expose.002	The Log Exposure Interface shall support the capability to support filtering of the logs.	Yes	Enables advanced log filtering and analysis through its Query DSL and various APIs such as /{index}/_search, /_search/scroll, /_search/scroll/<scroll-id>, /<index>/_msearch, /_msearch, making it easy to retrieve relevant logs based on specific criteria.
NOTE: Refer to the support of capabilities of log collection by the log aggregator function specified in clause 8.2 of ETSI GS NFV IFA 049 [1].			

A.2.3 Comparison of Log Analyser Interface requirements with considered open-source solutions capabilities

A.2.3.1 ElastAlert 2

This clause shows comparison of Log Analyser interface requirements defined in clause 6.2.5 of ETSI GS NFV IFA 049 [1] as "Identifier" column and "Requirement" column in Table A.2.3.1-1 and ElastAlert 2 as "Support by open source" and "Related capability of open source" column. The legend of "Support by open source" is the following:

"Yes": fully support the interface requirements.

"No": not support the interface requirements.

"Partial": partial support the interface requirements.

Table A.2.3.1-1: Comparison of Log Analyser interface requirements and ElastAlert 2

Identifier	Requirement	Support by open source	Related capability of open source
VNFLogAnalyser.001	The Log Analyser function shall support producing the Log Analysis Exposure Interface.	Yes	Utilizes custom alerting rules to analyze log data in OpenSearch or Elasticsearch and produce alert notifications, effectively serving as an analysis exposure interface. Alerts can be sent via email, custom webhooks, and other integrations, providing flexible delivery options.
LogAnalyser.Expose.001	The Log Analysis Exposure Interface shall support exposing the logs analysis results to authorized consumers.	Yes	Supports secure endpoints for delivering alerts, which can be customized to restrict access to authorized consumers only. This, combined with secure connection setups such as using TLS, TLS certificates or suppress related warnings, ensures that only intended consumers receive alerts.
LogAnalyser.Expose.002	The Log Analysis Exposure Interface shall support configuring the processing of logs to be analysed.	Yes	Enables detailed configuration of log processing through YAML rule files. These files allow the specification of data sources, definition of alert conditions, establishment of filtering criteria, and configuration of processing frequency for customized log analysis. This flexibility helps tailor the log analysis to specific needs and requirements.

A.2.3.2 Coroot

This clause shows comparison of Log Analyser interface requirements defined in clause 6.2.5 of ETSI GS NFV IFA 049 [1] as "Identifier" column and "Requirement" column in Table A.2.3.2-1 and Coroot as "Support by open source" and "Related capability of open source" column. The legend of "Support by open source" is the following:

"Yes": fully support the interface requirements.

"No": not support the interface requirements.

"Partial": partial support the interface requirements.

Table A.2.3.2-1: Comparison of Log Analyser interface requirements and Coroot

Identifier	Requirement	Support by open source	Related capability of open source
VNFLogAnalyser.001	The Log Analyser function shall support producing the Log Analysis Exposure Interface.	Yes	Uses the Coroot-node-agent and Coroot-cluster-agent to collect logs from various sources, such as containers, databases, based on the eBPF technology. Additionally, Coroot utilizes ClickHouse for storing logs, and benefiting from its efficient data compression technique. It also supports the OpenTelemetry protocol (OTLP over HTTP) for logs.
LogAnalyser.Expose.001	The Log Analysis Exposure Interface shall support exposing the logs analysis results to authorized consumers.	Yes	Supports integrations with various alerting platforms, including email, webhooks, and custom endpoints to notify authorized consumers of log analysis results. Additionally, configurations can be managed via the user interface, ensuring that access controls and notifications are appropriately set up.
LogAnalyser.Expose.002	The Log Analysis Exposure Interface shall support configuring the processing of logs to be analysed.	Yes	Dupports integrations with various alerting platforms, such as Incident Pages, webhooks, to notify authorized consumers of log analysis results. Additionally, configurations can be managed via the user interface, ensuring that access controls and notifications are appropriately set up.

A.2.3.3 Grafana®

This clause shows comparison of Log Analyser interface requirements defined in clause 6.2.5 of ETSI GS NFV IFA 049 [1] as "Identifier" column and "Requirement" column in Table A.2.3.3-1 and Grafana® as "Support by open source" and "Related capability of open source" column. The legend of "Support by open source" is the following:

"Yes": fully support the interface requirements.

"No": not support the interface requirements.

"Partial": partial support the interface requirements.

Table A.2.3.3-1: Comparison of Log Analyser interface requirements and Grafana®

Identifier	Requirement	Support by open source	Related capability of open source
VNFLogAnalyser.001	The Log Analyser function shall support producing the Log Analysis Exposure Interface.	Yes	Offers dynamic log visualization through integrations with Loki, Elasticsearch, Prometheus, and other data sources. It supports tools for visualizing and exploring log data through features like Grafana Explore and custom dashboards.
LogAnalyser.Expose.001	The Log Analysis Exposure Interface shall support exposing the logs analysis results to authorized consumers.	Yes	Supports this through its RBAC and API integrations. API endpoints such as the Dashboard API (/api/dashboards/), RBAC API (/api/access-control), Data source Permissions API (/api/access-control/datasources) enable secure access to logs and dashboards for authorized users, ensuring controlled exposure of log analysis results.
LogAnalyser.Expose.002	The Log Analysis Exposure Interface shall support configuring the processing of logs to be analysed.	Yes	In combination with Explore Logs, Loki or Grafana agent, allows configurable querying, filtering, and visualization. Users can set up data sources and define log processing rules, allowing for customized workflows. This also includes utilizing LogQL for querying and filtering logs dynamically.

A.2.3.4 OpenSearch

This clause shows comparison of Log Analyser interface requirements defined in clause 6.2.5 of ETSI GS NFV IFA 049 [1] as "Identifier" column and "Requirement" column in Table A.2.3.4-1 and OpenSearch as "Support by open source" and "Related capability of open source" column. The legend of "Support by open source" is the following:

"Yes": fully support the interface requirements.

"No": not support the interface requirements.

"Partial": partial support the interface requirements.

Table A.2.3.4-1: Comparison of Log Analyser interface requirements and OpenSearch

Identifier	Requirement	Support by open source	Related capability of open source
VNFLogAnalyser.001	The Log Analyser function shall support producing the Log Analysis Exposure Interface.	Yes	Support API endpoints such as Index API for ingesting log data and the Search API (/_search) to query and retrieve logs for analysis. Also allows the creation of dashboards via OpenSearch Dashboards, which can visualize log data. Additionally, users can query and analyse logs and present them in various formats such as charts and tables.
LogAnalyser.Expose.001	The Log Analysis Exposure Interface shall support exposing the logs analysis results to authorized consumers.	Yes	Supports built-in security features that enable RBAC, allows for the definition of roles and permissions to restrict access to log data and dashboards, ensuring only authorized users can view analysis results. Also supports API endpoints for this functionality include the Security Plugin for managing user roles and permissions, and the Security Plugin REST API (_plugins/_security/api/) to create and manage roles for user access control.
LogAnalyser.Expose.002	The Log Analysis Exposure Interface shall support configuring the processing of logs to be analysed.	Yes	Supports Ingest pipelines for preprocessing log data before indexing, but complex configurations may require additional queries. Also supports API endpoints include the Ingest Node for defining processing pipelines, the Ingest API (_ingest/pipeline) for managing these pipelines, and the Query DSL for building queries to analyse log data.

A.2.4 Comparison of Traffic Enforcer Interface requirements with considered open-source solutions capabilities

A.2.4.1 Cilium®

This clause shows comparison of Traffic Enforcer interface requirements defined in clause 6.2.1 of ETSI GS NFV IFA 049 [1] as "Identifier" column and "Requirement" column in Table A.2.4.1-1 and Cilium® as "Support by open source" and "Related capability of open source" column. The legend of "Support by open source" is the following:

"Yes": fully support the interface requirements.

"No": not support the interface requirements.

"Partial": partial support the interface requirements.

Table A.2.4.1-1: Comparison of Traffic Enforcer interface requirements and Cilium®

Identifier	Requirement	Support by open source	Related capability of open source
TrafficEnforcerInf .001	The Traffic Enforcer function shall support producing the traffic management Interface.	Yes	Supports various traffic management features, mainly through CiliumNetworkPolicy and CiliumClusterwideNetwork Policy, along with service mesh capabilities. It supports eBPF for efficient packet processing at the kernel level and supports technologies such as Kata Containers, Multi-Cluster (Cluster Mesh), ingress and egress gateways, the Cilium® agent, and both REST and gRPC APIs.
TrafficEnf.Trafm. 001	The Traffic Management Interface shall support the blocking and rerouting of traffic indicating selected VNFC Instances.	Yes	Supports capabilities for traffic blocking and rerouting through its network policy definitions. It allows for granular control of traffic to and from specific workloads (VNFC instances) using its CiliumNetworkPolicy and CiliumClusterwideNetwork Policy resources. Furthermore, it supports features like Egress Gateway, which can be configured to manage outbound traffic.

A.2.4.2 Istio®

This clause shows comparison of Traffic Enforcer interface requirements defined in clause 6.2.1 of ETSI GS NFV IFA 049 [1] as "Identifier" column and "Requirement" column in Table A.2.4.2-1 and Istio® as "Support by open source" and "Related capability of open source" column. The legend of "Support by open source" is the following:

"Yes": fully support the interface requirements.

"No": not support the interface requirements.

"Partial": partial support the interface requirements.

Table A.2.4.2-1: Comparison of Traffic Enforcer interface requirements and Istio®

Identifier	Requirement	Support by open source	Related capability of open source
TrafficEnforcerInf .001	The Traffic Enforcer function shall support producing the traffic management Interface.	Yes	Supports traffic management capabilities through configuration resources like VirtualService, allowing users to control routing behavior, including traffic splitting and circuit breaking, via YAML files. Components such as IstioOperator, and Istio® Gateway API, also support fine-grained traffic control, including retries and failover handling.
TrafficEnf.Trafm. 001	The Traffic Management Interface shall support the blocking and rerouting of traffic indicating selected VNFC Instances.	Yes	Supports traffic blocking through different components such as VirtualServices, Sidecar, which allows to define routing rules that either block traffic to specific services or reroute traffic based on various conditions (like HTTP headers, Ports, etc.).

A.2.4.3 Linkerd

This clause shows comparison of Traffic Enforcer interface requirements defined in clause 6.2.1 of ETSI GS NFV IFA 049 [1] as "Identifier" column and "Requirement" column in Table A.2.4.3-1 and Linkerd as "Support by open source" and "Related capability of open source" column. The legend of "Support by open source" is the following:

"Yes": fully support the interface requirements.

"No": not support the interface requirements.

"Partial": partial support the interface requirements.

Table A.2.4.3-1: Comparison of Traffic Enforcer interface requirements and Linkerd

Identifier	Requirement	Support by open source	Related capability of open source
TrafficEnforcerInf .001	The Traffic Enforcer function shall support producing the traffic management Interface.	Yes	Supports traffic management through features like Traffic Split, enabling efficient traffic routing. It exposes this capability via the SMI TrafficSplit API, allowing for traffic distribution across various application versions or instances, facilitating canary releases and blue/green deployments.
TrafficEnf.Trafm. 001	The Traffic Management Interface shall support the blocking and rerouting of traffic indicating selected VNFC Instances.	Partial	Partially supports this requirement by enabling rerouting and flow control policies. While it does not provide explicit blocking controls for individual instances, users can effectively manage traffic using policies, annotations in Service Profiles, and service mappings. Additionally, features like Circuit Breaking feature can isolate failing services by stopping traffic to them temporarily, while Rate Limiting feature can control the volume of requests sent to specific services. Together, these capabilities can enhance traffic management and resilience within the mesh.

A.2.4.4 Envoy

This clause shows comparison of Traffic Enforcer interface requirements defined in clause 6.2.1 of ETSI GS NFV IFA 049 [1] as "Identifier" column and "Requirement" column in Table A.2.4.4-1 and Envoy as "Support by open source" and "Related capability of open source" column. The legend of "Support by open source" is the following:

"Yes": fully support the interface requirements.

"No": not support the interface requirements.

"Partial": partial support the interface requirements.

Table A.2.4.4-1: Comparison of Traffic Enforcer interface requirements and Envoy

Identifier	Requirement	Support by open source	Related capability of open source
TrafficEnforcerInf .001	The Traffic Enforcer function shall support producing the traffic management Interface.	Yes	Supports traffic management through its routing capabilities. It allows the definition of routes that determine how traffic is handled based on various criteria (e.g. headers, paths). The Envoy API facilitates dynamic management of these routes; API endpoints such as v3.Route, v3.Listener, v3.RouteAction, v3.Filter provide mechanisms for configuring routing behaviour, managing incoming traffic, specifying traffic actions, and implementing custom request and response logic.
TrafficEnf.Trafm. 001	The Traffic Management Interface shall support the blocking and rerouting of traffic indicating selected VNFC Instances.	Partial	Supports rerouting and blocking of traffic through features like Route Matching and HTTPFilters, which enable custom logic for managing traffic flows. Endpoints such as v3.RouteAction defines specific actions for rerouting traffic, and v3.Filter allows for the implementation of custom filtering logic. Although Envoy can block and reroute traffic based on various criteria, directly indicating instances may require additional custom logic or tagging to fully implement this requirement.

A.2.5 Comparison of Notification Manager Interface requirements with considered open-source solutions capabilities

A.2.5.1 Prometheus Alertmanager

This clause shows comparison of Notification manager interface requirements defined in clause 6.2.11 of ETSI GS NFV IFA 049 [1] (see "Identifier" column and "Requirement" column in Table A.2.5.1-1) and Prometheus Alertmanager (see "Support by open source" and "Related capability of open source" column). The legend of "Support by open source" is the following:

"Yes": fully support the interface requirements.

"No": not support the interface requirements.

"Partial": partial support the interface requirements.

Table A.2.5.1-1: Comparison of Notification interface requirements and Prometheus Alertmanager

Identifier	Requirement	Support by open source	Related capability of open source
NotificationManagerInf .001	The Notification Manager function shall support producing the Notifications Management Interface.	Partial	It exposes a management interface through its rest API for managing alerts, silences, and status monitoring. It also provides a Web UI for visualization and operations. However, it lacks a dedicated interface for managing notifications dynamically.
Notif.Manager Notif.Mgmt.001	The Notifications Management Interface shall support management of subscriptions to notifications.	Partial	Allows configuration-based routing of alerts but does not have a built-in API for managing dynamic subscriptions. External automation tools are needed.
Notif.Manager Notif.Mgmt.002	The Notifications Management Interface shall support sending processed notifications to authorized consumers.	Yes	Processes and routes alerts to various consumers.

A.2.5.2 Argo®

This clause shows comparison of Notification Manager interface requirements defined in clause 6.2.11 of ETSI GS NFV IFA 049 [1] (see "Identifier" column and "Requirement" column in Table A.2.5.2-1) and Argo® (see "Support by open source" and "Related capability of open source" column). The legend of "Support by open source" is the following:

"Yes": fully support the interface requirements.

"No": not support the interface requirements.

"Partial": partial support the interface requirements.

Table A.2.5.2-1: Comparison of Notification Manager interface requirements and Argo®

Identifier	Requirement	Support by open source	Related capability of open source
NotificationManagerInf .001	The Notification Manager function shall support producing the Notifications Management Interface.	Partial	It does not provide a native notification management interface. Argo® CD Notifications Controller, provides a framework for sending notifications using predefined or custom templates.
Notif.Manager Notif.Mgmt.001	The Notifications Management Interface shall support management of subscriptions to notifications.	Partial	Subscription to events is supported, but un-subscription and dynamic subscription lifecycle management are limited.
Notif.Manager Notif.Mgmt.002	The Notifications Management Interface shall support sending processed notifications to authorized consumers.	Yes	Support sending notifications to designated consumers.

A.2.5.3 Kafka

This clause shows comparison of Notification manager interface requirements defined in clause 6.2.11 of ETSI GS NFV IFA 049 [1] (see "Identifier" column and "Requirement" column in Table A.2.5.3-1) and Kafka (see "Support by open source" and "Related capability of open source" column). The legend of "Support by open source" is the following:

"Yes": fully support the interface requirements.

"No": not support the interface requirements.

"Partial": partial support the interface requirements.

Table A.2.5.3-1: Comparison of Notification Manager interface requirements and Kafka

Identifier	Requirement	Support by open source	Related capability of open source
NotificationManagerInf .001	The Notification Manager function shall support producing the Notifications Management Interface.	Yes	Kafka supports notification management through its publish-subscribe model. Producers publish notifications to topics, and consumers can subscribe to receive them. Kafka's Streams API and Connect API further enable processing, routing, and management of notifications.
Notif.Manager Notif.Mgmt.001	The Notifications Management Interface shall support management of subscriptions to notifications.	Partial	It does not provide a native API for managing consumer subscriptions dynamically (e.g. listing, modifying, or deleting specific subscriptions per user or service). However, supports dynamic subscription management using Kafka Consumer Groups, where consumers can join or leave at runtime. Kafka Admin API and Kafka REST Proxy allow programmatic subscription management.
Notif.Manager Notif.Mgmt.002	The Notifications Management Interface shall support sending processed notifications to authorized consumers.	Yes	Provides message retention and supports querying notification status and history using Kafka Connect with external storage (Elasticsearch, Prometheus, PostgreSQL, etc.). Kafka consumer offsets allow tracking delivery status, and failure logs can be handled using dead-letter topics.

A.2.5.4 Sensu

This clause shows comparison of Notification manager interface requirements defined in clause 6.2.11 of ETSI GS NFV IFA 049 [1] (see "Identifier" column and "Requirement" column in Table A.2.5.4-1) and Sensu (see "Support by open source" and "Related capability of open source" column). The legend of "Support by open source" is the following:

"Yes": fully support the interface requirements.

"No": not support the interface requirements.

"Partial": partial support the interface requirements.

Table A.2.5.4-1: Comparison of Notification Manager interface requirements and Sensu

Identifier	Requirement	Support by open source	Related capability of open source
NotificationManagerInf .001	The Notification Manager function shall support producing the Notifications Management Interface.	Yes	Supports notification management using its event pipeline. It can process events from multiple sources, apply filtering, and send notifications to defined handlers. Sensu's API and CLI offer interfaces for managing notifications and integrating with external systems.
Notif.Manager Notif.Mgmt.001	The Notifications Management Interface shall support management of subscriptions to notifications.	Partial	Supports subscription-based event routing via configuration and API settings. Consumers (agents) can subscribe to event sources, but subscriptions are typically defined in advance and are not dynamically managed at runtime like a full pub/sub system.
Notif.Manager Notif.Mgmt.002	The Notifications Management Interface shall support sending processed notifications to authorized consumers.	Partial	Supports sending processed notifications to authorized consumers through handlers, RBAC, and subscriptions. Notifications are processed in real time but not stored for later retrieval.

A.2.6 Comparison of VNF Metrics Aggregator Interface requirements with considered open-source solutions capabilities

A.2.6.1 OpenTelemetry Collector

This clause shows comparison of VNF Metrics Aggregator interface requirements defined in clause 6.2.6 of ETSI GS NFV IFA 049 [1] (see "Identifier" column and "Requirement" column in Table A.2.6.1-1) and OpenTelemetry Collector (see "Support by open source" and "Related capability of open source" column). The legend of "Support by open source" is the following:

"Yes": fully support the interface requirements.

"No": not support the interface requirements.

"Partial": partial support the interface requirements.

Table A.2.6.1-1: Comparison of VNF Metrics Aggregator interface requirements and OpenTelemetry Collector

Identifier	Requirement	Support by open source	Related capability of open source
VNFMetricAggregator.001	The VNF Metrics Aggregator function shall support producing the Metrics Exposure Interface (see note).	Yes	Receivers can be configured to collect metrics via file-based (e.g. Prometheus, OTLP JSON File receiver) or streaming-based mechanisms (OTLP, Kafka receiver) allowing for effective metrics aggregation.
MetricAggr.Expose.001	The Metrics Exposure Interface shall support exposing the metrics to authorized consumers.	Yes	Supports TLS/mTLS for secure communication and mutual authentication options to secure transmission of metrics. Authenticators (e.g. OIDC authenticator) can also be integrated to ensure that only authorized requests are processed.
MetricAggr.Expose.002	The Metrics Exposure Interface shall support the capability to support filtering of the metrics.	Yes	Processors support the filtering and the transformation of metrics, allowing users to manage which metrics are exposed based on specific criteria.
NOTE: Refer to the support of capabilities of metrics collection by the metrics aggregator function specified in clause 8.2 of ETSI GS NFV-IFA 049 [1].			

A.2.7 Comparison of VNF Metrics Analyser Interface requirements with considered open-source solutions capabilities

A.2.7.1 Coroot

This clause shows comparison of VNF Metrics Analyser interface requirements defined in clause 6.2.7 of ETSI GS NFV IFA 049 [1] as "Identifier" column and "Requirement" column in Table A.2.7.1-1 and Coroot as "Support by open source" and "Related capability of open source" column. The legend of "Support by open source" is the following:

"Yes": fully support the interface requirements.

"No": not support the interface requirements.

"Partial": partial support the interface requirements.

Table A.2.7.1-1: Comparison of VNF Metrics Analyser interface requirements and Coroot

Identifier	Requirement	Support by open source	Related capability of open source
VNFMetricAnalyser.001	The VNF Metrics Analyser function shall support producing the Metrics Analysis Exposure Interface.	Yes	Supports observability by collecting and analyzing telemetry data, including metrics, logs, traces, and profiles, using eBPF technology. Provides dashboards to display and analyse the collected metrics efficiently.
MetricAnalyser.Expose.001	The Metrics Analysis Exposure Interface shall support exposing the metrics analysis results to authorized consumers.	Partial	Provides RBAC for managing access to metrics analysis results, though access control capabilities may be limited in some configurations. Metrics analysis results can be accessed via dashboards and shared through notifications via webhooks, and other channels.
MetricAnalyser.Expose.002	The Metrics Analysis Exposure Interface shall support configuring the processing of metrics to be analysed.	Partial	Supports configuration of metrics processing by allowing users to define thresholds for inspections and set SLOs. However, customization is limited to predefined parameters and system configurations, without full flexibility for advanced metric processing.

A.2.7.2 OpenSearch

This clause shows comparison of VNF Metrics Analyser interface requirements defined in clause 6.2.7 of ETSI GS NFV IFA 049 [1] as "Identifier" column and "Requirement" column in Table A.2.7.2-1 and OpenSearch as "Support by open source" and "Related capability of open source" column. The legend of "Support by open source" is the following:

"Yes": fully support the interface requirements.

"No": not support the interface requirements.

"Partial": partial support the interface requirements.

Table A.2.7.2-1: Comparison of VNF Metrics Analyser interface requirements and OpenSearch

Identifier	Requirement	Support by open source	Related capability of open source
VNFMetricAnalyser.001	The VNF Metrics Analyser function shall support producing the Metrics Analysis Exposure Interface.	Yes	Offers capabilities for producing a Metrics Analysis Exposure Interface. It allows users to collect, process, and visualize metrics through various tools and plugins. For instance, OpenSearch can ingest metrics from sources like Prometheus and OpenTelemetry, enabling analysis and visualization within OpenSearch Dashboards.
MetricAnalyser.Expose.001	The Metrics Analysis Exposure Interface shall support exposing the metrics analysis results to authorized consumers.	Yes	Ensures that metrics analysis results are accessible to authorized consumers through its integration with OpenSearch Dashboards. Users can create custom visualizations and dashboards to display metrics data. Additionally, OpenSearch supports RBAC, ensuring that only authorized users can access specific metrics and dashboards.
MetricAnalyser.Expose.002	The Metrics Analysis Exposure Interface shall support configuring the processing of metrics to be analysed.	Yes	Provides capabilities for configuring metric processing. Users can define structured queries, implement transformation rules, and set up data pipelines to control metric ingestion, aggregation, and storage. It supports real-time data processing and customizable workflows, ensuring efficient handling of metrics for analysis and visualization.

A.2.7.3 VictoriaMetrics

This clause shows comparison of VNF Metrics Analyser interface requirements defined in clause 6.2.7 of ETSI GS NFV-IFA 049 [1] as "Identifier" column and "Requirement" column in Table A.2.7.3-1 and VictoriaMetrics as "Support by open source" and "Related capability of open source" column. The legend of "Support by open source" is the following:

"Yes": fully support the interface requirements.

"No": not support the interface requirements.

"Partial": partial support the interface requirements.

Table A.2.7.3-1: Comparison of VNF Metrics Analyser interface requirements and VictoriaMetrics

Identifier	Requirement	Support by open source	Related capability of open source
VNFMetricAnalyser.001	The VNF Metrics Analyser function shall support producing the Metrics Analysis Exposure Interface.	Yes	Exposes analysis results such as metrics from recording rules and alert statuses through its Query API, serving as an exposure interface for analytics.
MetricAnalyser.Expose.001	The Metrics Analysis Exposure Interface shall support exposing the metrics analysis results to authorized consumers.	Partial	Exposes analytics results via HTTP endpoints. However, authentication and authorization mechanisms (e.g. OAuth2, RBAC) can be implemented externally using reverse proxies, or API gateways, as VictoriaMetrics does not support these natively.
MetricAnalyser.Expose.002	The Metrics Analysis Exposure Interface shall support configuring the processing of metrics to be analysed.	Partial	Supports rule-based configuration through VMRule defined recording and alerting rules, providing the capability to define thresholds and compose analytic expressions using MetricsQL. However, lacks advanced pipeline management features or runtime configurability beyond static rule definitions.

A.3 Comparison of the considered open-source solutions against VNF generic OAM functions' functional and interface requirements

A.3.1 Comparison of the considered open-source solutions against Log Aggregator functional and interface requirements

Table A.3.1-1 and Table A.3.1-2 provide a comparison of open source solutions against the Log Aggregator functional and interface requirements, based on the "Requirements" columns in:

- Tables A.1.2.1.2-1 to A.1.2.5.2-1 for the functional requirements, with details listed for each requirement identifier.
- Tables A.2.2.1-1 to A.2.2.5-1 for the interface requirements, with details listed for each requirement identifier.

The legend of "Support by open-source" is the following:

"Yes": fully support the functional/interface requirements.

"No": not support the functional/interface requirements.

"Partial": partial support the functional/interface requirements.

Table A.3.1-1: Comparison of the open-source solutions against the Log Aggregator functional requirements

Identifier	Support by Fluent Bit	Support by Fluent	Support by OpenTelemetry Collector	Support by Grafana Loki	Support by OpenSearch
LogAggregator.001	Yes	Yes	Yes	Yes	Yes
LogAggregator.002	Yes	Yes	Yes	No	Partial
LogAggregator.003	Yes	Yes	Yes	No	Yes
LogAggregator.004	Yes	Yes	Yes	Yes	Yes
LogAggregator.005	Yes	Yes	Yes	Yes	Yes

Table A.3.1-2: Comparison of the open-source solutions against the Log Aggregator interface requirements

Identifier	Support by Fluent Bit	Support by Fluent	Support by OpenTelemetry Collector	Support by Grafana Loki	Support by OpenSearch
VNFLogAggregator.001	Yes	Yes	Yes	Yes	Yes
LogAggr.Expose.001	Yes	Yes	Yes	Yes	Yes
LogAggr.Expose.002	Yes	Yes	Yes	Yes	Yes

A.3.2 Comparison of the considered open-source solutions against Log Analyser functional and interface requirements

Table A.3.2-1 and Table A.3.2-2 provide a comparison of open source solutions against the Log Analyser functional and interface requirements, based on the "Requirements" columns in:

- Tables A.1.3.1.2-1 to A.1.3.4.2-1 for the functional requirements, with details listed for each requirement identifier.
- Tables A.2.3.1-1 to A.2.3.4-1 for the interface requirements, with details listed for each requirement identifier.

The legend of "Support by open-source" is the following:

"Yes": fully support the functional/interface requirements.

"No": not support the functional/interface requirements.

"Partial": partial support the functional/interface requirements.

Table A.3.2-1: Comparison of the open-source solutions against the Log Analyser functional requirements

Identifier	Support by ElastAlert 2	Support by Coroot	Support by Grafana®	Support by OpenSearch
LogAnalyser.001	Yes	Yes	Yes	Yes
LogAnalyser.002	Yes	Yes	Yes	Yes
LogAnalyser.003	Yes	Yes	Yes	Yes
LogAnalyser.004	Yes	Yes	Yes	Yes

Table A.3.2-2: Comparison of the open-source solutions against the Log Analyser interface requirements

Identifier	Support by ElastAlert 2	Support by Coroot	Support by Grafana®	Support by OpenSearch
VNFLogAnalyser.001	Yes	Yes	Yes	Yes
LogAnalyser.Expose.001	Yes	Yes	Yes	Yes
LogAnalyser.Expose.002	Yes	Yes	Yes	Yes

A.3.3 Comparison of the considered open-source solutions against Traffic Enforcer functional and interface requirements

Table A.3.3-1 and Table A.3.3-2 provide a comparison of open source solutions against the Traffic Enforcer functional and interface requirements, based on the "Requirements" columns in:

- Tables A.1.4.1.2-1 to A.1.4.4.2-1 for the functional requirements, with details listed for each requirement identifier.
- Tables A.2.4.1-1 to A.2.4.4-1 for the interface requirements, with details listed for each requirement identifier.

The legend of "Support by open-source" is the following:

"Yes": fully support the functional/interface requirements.

"No": not support the functional/interface requirements.

"Partial": partial support the functional/interface requirements.

Table A.3.3-1: Comparison of the open-source solutions against the Traffic Enforcer functional requirements

Identifier	Support by Cilium®	Support by Istio®	Support by Linkerd	Support by Envoy
TrafficEnforcer.001	Yes	Yes	Yes	Yes

Table A.3.3-2: Comparison of the open-source solutions against the Traffic Enforcer functional and interface requirements

Identifier	Support by Cilium®	Support by Istio®	Support by Linkerd	Support by Envoy
TrafficEnforcerInf.001	Yes	Yes	Yes	Yes
TrafficEnf.Trafm.001	Yes	Yes	Partial	Partial

A.3.4 Comparison of the considered open-source solutions against Notification Manager functional and interface requirements

Table A.3.4-1 and Table A.3.4-2 provide a comparison of open source solutions against the Notification Manager functional and interface requirements, based on the "Requirements" columns in:

- Tables A.1.8.1.2-1 to A.1.8.4.2-1 for the functional requirements, with details listed for each requirement identifier.
- Tables A.2.5.1-1 to A.2.5.4-1 for the interface requirements, with details listed for each requirement identifier.

The legend of "Support by open-source" is the following:

"Yes": fully support the functional/interface requirements.

"No": not support the functional/interface requirements.

"Partial": partial support the functional/interface requirements.

Table A.3.4-1: Comparison of the open-source solutions against the Notification Manager functional requirements

Identifier	Support by Prometheus Alertmanager	Support by Argo®	Support by Kafka	Support by Sensu
NotificationManager.001	Yes	Partial	Yes	Yes
NotificationManager.002	Yes	Partial	Partial	Partial
NotificationManager.003	Yes	Yes	Yes	Yes
NotificationManager.004	Yes	Partial	Partial	Partial

Table A.3.4-2: Comparison of the open-source solutions against the Notification Manager interface requirements

Identifier	Support by Prometheus Alertmanager	Support by Argo®	Support by Kafka	Support by Sensu
NotificationManagerInf.001	Partial	Partial	Yes	Yes
Notif.Manager Notif.Mgmt.001	Partial	Partial	Partial	Partial
Notif.Manager Notif.Mgmt.002	Yes	Yes	Yes	Partial

A.3.5 Comparison of the considered open-source solutions against VNF Metrics Analyser functional and interface requirements

Table A.3.5-1 and Table A.3.5-2 provide a comparison of open source solutions against the Metrics Analyser functional and interface requirements, based on the "Requirements" columns in:

- Tables A.1.7.1.2-1 to A.1.7.3.2-1 for the functional requirements, with details listed for each requirement identifier.
- Tables A.2.7.1-1 to A.2.7.3-1 for the interface requirements, with details listed for each requirement identifier.

The legend of "Support by open-source" is the following:

"Yes": fully support the functional/interface requirements.

"No": not support the functional/interface requirements.

"Partial": partial support the functional/interface requirements

Table A.3.5-1: Comparison of the open-source solutions against the VNF Metrics Analyser functional requirements

Identifier	Support by Coroot	Support by OpenSearch	Support by VictoriaMetrics
VNFMetricAnalyser.001	Yes	Yes	Partial
VNFMetricAnalyser.002	Partial	Yes	Partial
VNFMetricAnalyser.003	Partial	Yes	Yes
VNFMetricAnalyser.004	Yes	Yes	Partial

Table A.3.5-2: Comparison of the open-source solutions against the VNF Metrics Analyser functional and interface requirements

Identifier	Support by Coroot	Support by OpenSearch	Support by VictoriaMetrics
VNFMetricAnalyser.001	Yes	Yes	Yes
MetricAnalyser.Expose.001	Partial	Yes	Partial
MetricAnalyser.Expose.002	Partial	Yes	Partial

A.4 Cross-comparison of considered open-source solutions

A.4.1 Cross-comparison of open-source solutions for Log Aggregator Function

Based on the solutions considered in clause A.1.2, Table A.4.1-1 shows a cross-comparison of the open-source solutions (FluentBit, Fluentd, OpenTelemetry Collector, Grafana Loki, and OpenSearch) based on defined criteria focused on functional, interface, and non-functional requirements.

Table A.4.1-1: Cross-comparison of open-source solutions for Log Aggregator Function

Cross comparison criteria	Fluent Bit	Fluentd	OpenTelemery Collector	Grafana Loki	OpenSearch
LogAggregator.001	Collects logs from various sources including VNF instances, applications, and infrastructure with flexible filtering capabilities.	Collects logs from various sources with support for flexible filtering based on log attributes.	Capable of collecting logs from various sources, supporting filtering through processor components.	Collects logs via integrations with Fluentd or Fluent Bit; filtering is possible during querying.	Capable of collecting logs via Logstash or Beats, supports filtering for log types and severity.
LogAggregator.002	Provides log parsing and format conversion and supports basic enrichment functionalities.	Offers filtering and parsing plugins to standardize log formats and modify records as needed.	Can pre-process logs with processors for format conversion and enrichment.	Does not provide built-in pre-processing; requires upstream tools for any format conversion.	Integrates with Logstash/Beats for pre-processing, allowing formatting, filtering, and enriching logs.
LogAggregator.003	Supports configurable log aggregation and routing based on various criteria.	Provides output plugins for aggregation based on log level and instance type, with buffering for efficiency.	Enables configurable log aggregation using flexible pipelines.	Does not perform advanced log aggregation; querying uses Loki's query language for aggregation.	Supports configurable aggregation through Logstash pipelines or query-time aggregation.
LogAggregator.004	Forwards logs to external storage systems for long-term storage.	Forwards logs to external systems or vendor-specific backends for long-term storage and historical retention.	Forwards logs to external systems that manage historical retention, like Prometheus or vendor-specific backends.	Stores historical logs with a scalable backend optimized for managing large volumes.	Capable of storing logs for long-term retention, supports time-based indices for efficient management of large volumes of log records.

Cross comparison criteria	Fluent Bit	Fluentd	OpenTelemery Collector	Grafana Loki	OpenSearch
LogAggregator.005	Forwards filtered logs to systems that provide exposure functionalities.	Forwards filtered logs to external systems that expose logs to authorized consumers.	Forwards logs to systems enabling access controls for exposing logs to authorized consumers.	Integrates with Grafana® for visualization and querying; access control managed through Grafana® permissions.	Offers RBAC to expose filtered logs, visualized using OpenSearch Dashboards.
VNFLogAggregator.001	Supports multiple output plugins for log exposure.	Supports various output plugins for log exposure to multiple destinations.	Supports various receivers to ingest logs, allowing effective log aggregation.	Can ingest logs from various sources through Loki API and Promtail.	Supports ingest APIs for batch log ingestion, allowing logs to be sent directly to OpenSearch.
LogAggr.Expose.001	Supports secure log transmission with TLS/SSL encryption and Basic Authentication.	Supports secure log transmission and various output plugins for authorization.	Supports secure transport and authentication options for HTTP receivers.	Integrates with Grafana® over TLS for user management and access control.	Offers robust security features ensuring that only authorized users can access logs.
LogAggr.Expose.002	Supports various filter plugins for customizable log exposure.	Supports various filter plugins to tailor log exposure based on specific criteria.	Processor supports filtering and modifying logs for tailored exposure.	Supports complex queries using LogQL for effective log management.	Enables advanced log filtering and analysis through Query DSL and various APIs.
CRD Support	Supports Custom Resource Definitions for Kubernetes®.	Supports CRDs for Kubernetes® integrations.	Supports Kubernetes® CRDs for integrating with various observability tools.	Does not directly support CRDs but can be used in Kubernetes® environments.	Supports CRDs for integrating with Kubernetes®, enabling customization for log management workflows.
Community and Ecosystem	Established community support with numerous integrations and plugins available.	Large community with extensive plugins and integrations available, widely adopted in production.	Growing community; integrates well with existing observability tools.	Active community, particularly in combination with Grafana®, widely used in observability setups.	Strong ecosystem with support from Amazon, extensive documentation, and various plugins available.
Scalability	Designed to handle high log throughput in low-resource environments.	Scalable solution for large log volumes, capable of clustering for high availability.	Capable of efficiently handling large volumes of telemetry data through its architecture.	Scalable storage backend that manages large volumes of log data effectively.	Capable of managing large volumes of log data through horizontal scaling and distributed architecture.
Customization and Extensibility	Customizable with a range of plugins for output, filters, and format conversions.	Extensible with support for custom plugins and configurations for diverse log handling needs.	Extensible through various components, allowing users to create custom pipelines and processes.	Limited extensibility; primarily focused on log storage and querying rather than extensive customization.	Customizable with support for various data ingestion methods, plugins, and APIs for tailored log management.

Based on this comparison, among the open-source solution candidates Fluent Bit, Fluentd, OpenTelemetry Collector, Grafana Loki, and OpenSearch, no single tool stands out as the absolute best; instead, the choice depends on specific use cases and functionality requirements.

Based on the overall analysis in clauses A.1.2, A.2.2, and A.4.1, most Log Aggregator functional and interface requirements can be met by the considered open-source tools. However, Fluent Bit and OpenTelemetry Collector stand out as the most suitable options. The choice ultimately depends on the specific use cases and functionalities required. For example, Fluent Bit is lightweight and optimized for high-throughput environments, making it ideal for low-latency log processing in Kubernetes® or resource-constrained environments. On the other hand, OpenTelemetry Collector's multi-signal support for logs, metrics, and traces makes it preferable for environments prioritizing comprehensive observability and detailed data aggregation across diverse sources.

A.4.2 Cross-comparison of open-source solutions for Log Analyser Function

Based on the solutions considered in clause A.1.3, Table A.4.2-1 shows a cross-comparison of the open-source solutions (ElastAlert 2, Coroot, Grafana®, and OpenSearch) based on defined criteria focused on functional, interface, and non-functional requirements.

Table A.4.2-1: Cross-comparison of open-source solutions for Log Analyser Function

Cross comparison criteria	ElastAlert 2	Coroot	Grafana®	OpenSearch
LogAnalyser.001	Supports functions such as abnormal behaviour detection (including spikes, flatlines, blacklist, whitelist), threshold crossing, and basic statistical processing.	Capable of detecting abnormal behaviour, including performance issues and resource bottlenecks in applications.	Integrates with Loki for logs, enabling analysis such as threshold crossing, abnormal behaviour detection, and visual correlation of logs.	Facilitates log analysis functions, including anomaly detection, threshold-based alerting, statistical processing, and log correlation.
LogAnalyser.002	Configurable through YAML files, allowing the setting of thresholds, defining time windows, combining rules, and customizing analysis workflows.	Provides options to configure alerting thresholds and customize log monitoring settings to track key performance metrics.	Allows users to configure dashboards with thresholds, queries, and alerts, relying on external log processing tools for core analytics.	Configuration available through alerting features, offering customizable thresholds, time windows, and rule-based processing for log data.
LogAnalyser.003	Capable of sending notifications via email, relevant platforms, and custom scripts based on alerting conditions.	Integrates with various platforms to notify users of detected issues or alerts based on analysis findings.	Supports alerts and notifications through integrations with relevant platforms, triggered by custom conditions applied to data (e.g. log errors, thresholds).	Capable of sending notifications through an alerting framework, integrating with email, webhooks, and relevant platforms based on log analysis.
LogAnalyser.004	Can send alerts to external systems that expose results, although lacks a native API or dashboard for analytics; custom integrations required for exposing analytics results.	Provides dashboards and visual reports for easy access and sharing of analytics data with authorized users.	Offers RBAC for securely sharing dashboards and analytics results with authorized users through its web-based interface.	OpenSearch Dashboards provide visual analytics results, shared securely with authorized consumers via roles and access controls.
VNFLogAnalyser.001	Utilizes custom alerting rules to analyze log data in OpenSearch or Elasticsearch, generating alert notifications.	Uses Coroot-node-agent and Coroot-cluster-agent to collect logs, with support for the OpenTelemetry protocol for logs.	Offers dynamic log visualization through integrations with Loki, Elasticsearch, and Prometheus.	Supports API endpoints for ingesting log data and querying it, allowing the creation of dashboards for visualizing log data.

Cross comparison criteria	ElastAlert 2	Coroot	Grafana®	OpenSearch
LogAnalyser.Expose.001	Supports secure endpoints for delivering alerts, ensuring alerts reach intended consumers.	Supports integrations with various alerting platforms, ensuring access controls are properly configured.	Enables controlled exposure of log analysis results through RBAC and API integrations.	Offers built-in security features that facilitate RBAC, ensuring only authorized users can view analysis results.
LogAnalyser.Expose.002	Enables detailed configuration of log processing through YAML rule files for tailored analysis.	Supports integrations with alerting platforms and offers user interface configurations for access controls and notifications.	Allows configurable querying and filtering in combination with Explore Logs, facilitating customized workflows.	Provides support for ingest pipelines that preprocess log data, with API endpoints available for defining processing pipelines and building queries.
CRD Support	Lacks direct support, requiring custom implementations for integration.	Offers support for CRD via Kubernetes® integration focused on monitoring.	Not applicable directly; integrates with external data sources.	Provides support for CRD through OpenSearch features and plugins for customized setups.
Community and Ecosystem	Maintains an active community with ongoing development efforts.	Developing ecosystem with integrations concentrated on performance monitoring.	Benefits from a large community and extensive plugin ecosystem for visualization.	Features an active community with numerous plugins and ongoing updates.
Scalability	Scales efficiently with horizontal scaling options based on Elasticsearch.	Architecture designed for scalability utilizing ClickHouse for log storage and eBPF technology.	Demonstrates scalability, leveraging backend data sources for managing large datasets.	Architecture capable of handling large datasets and supporting distributed log processing.
Customization and Extensibility	Offers customization through YAML configurations but may require additional scripting for complex use cases.	Customization options available through agent configurations and user interface settings.	Extensive customization options available through dashboard configuration and plugin support.	Provides robust customization through APIs and query DSL, enabling tailored log processing and analysis.

Based on this comparison, among open-source solution candidates ElastAlert 2, Coroot, Grafana®, and OpenSearch, no single tool stands out as the absolute best; instead, the choice depends on specific use cases and functionality requirements.

Based on the overall analysis in clauses A.1.3, A.2.3, and A.4.2, most Log Analyser functional and interface requirements can be met by the considered open-source tools. However, OpenSearch and Grafana® stand out as the most suitable options. The choice ultimately depends on the specific use cases and functionalities required. For example, OpenSearch offers robust search capabilities and advanced query features, making it ideal for detailed log analysis, visualization, and real-time monitoring. On the other hand, Grafana® excels in creating dynamic dashboards that facilitate visual analytics and comprehensive insights from log data.

A.4.3 Cross-comparison of open-source solutions for Traffic Enforcer Function

Based on the solutions considered in clause A.1.4, Table A.4.3-1 shows a cross-comparison of the open-source solutions (Cilium®, Istio®, Linkerd, and Envoy) based on defined criteria focused on functional, interface, and non-functional requirements.

Table A.4.3-1: Cross-comparison of open-source solutions for Traffic Enforcer Function

Cross comparison criteria	Cilium®	Istio®	Linkerd	Envoy
TrafficEnforcer.001	Supports traffic isolation and rerouting using eBPF-based policies, enabling control over microservices communication.	Allows traffic isolation with policies for managing communication between microservices. Traffic rerouting is facilitated through virtual services and destination rules.	Implements traffic policies with rate limiting and circuit breaking, enabling partial isolation and rerouting through load balancing and failover features.	Offers traffic isolation and rerouting through routing and service mesh features, incorporating rate limiting and traffic management across microservices.
TrafficEnforcerInf.001	Produces the traffic management interface via CiliumNetworkPolicy and CiliumClusterwideNetworkPolicy. Integration with ingress/egress gateways and both REST and gRPC APIs is supported.	Provides traffic management capabilities through configuration resources like VirtualService. Users can control routing behavior via YAML files, supporting retries and failover handling.	Facilitates traffic management through features like Traffic Split, which allows for efficient routing and canary releases using the SMI, TrafficSplit API.	Enables traffic management via routing capabilities, allowing the definition of routes based on various criteria, with dynamic management facilitated through the Envoy API.
TrafficEnf.Trafm.001	Allows for blocking and rerouting traffic using network policy definitions, providing granular control of workloads instances. The Egress Gateway can manage outbound traffic.	Supports traffic blocking and rerouting using VirtualServices and Sidecar, which define rules for managing traffic flow to specific services based on various conditions.	Provides capabilities for rerouting and flow control policies, with features such as circuit breaking to isolate failing services and rate limiting for traffic control.	Facilitates rerouting and blocking of traffic through features like Route Matching and HTTPFilters, which enable custom logic for managing traffic flows. Custom logic may be needed for instance-based control.
CRD Support	Strong support for CRDs via Kubernetes® for traffic policies, enhancing integration and extensibility.	Comprehensive CRD support through Istio®'s resources, such as VirtualService, DestinationRule, and Gateway, enabling detailed traffic management configurations.	Limited CRD support, primarily through Service Profiles and TrafficSplit configurations within the SMI.	Good support for custom resources, primarily focused on service definitions and routing configurations rather than dedicated traffic management CRDs.
Community and Ecosystem	A growing community with active contributions and support for various integrations, particularly within Kubernetes® environments.	Well-established ecosystem with extensive documentation and community support. Strong integrations with Kubernetes® and various cloud-native tools enhance its usability.	An emerging community with increasing adoption and integration with Kubernetes®, supporting service mesh patterns.	A robust community and ecosystem that is widely used across various cloud-native applications, offering strong support for integrations with existing systems and tools.
Scalability	Highly scalable due to its eBPF architecture, allowing efficient packet processing without significant overhead, making it suitable for large environments.	Scalable architecture supports large deployments, with fine-grained traffic control and routing policies to effectively manage extensive microservice interactions.	Offers good scalability, with features like Traffic Split for efficient distribution and control of traffic across multiple service versions.	Highly scalable and performant, capable of managing large volumes of traffic with low latency due to its efficient architecture and service mesh capabilities.

Cross comparison criteria	Cilium®	Istio®	Linkerd	Envoy
Customization and Extensibility	Highly customizable with support for various policies and configurations through YAML files and eBPF, enabling tailored traffic management solutions.	Extensive customization options are available through configurations in YAML and custom resource definitions, allowing users to define specific routing and traffic management behaviors.	Customization is achievable, particularly in traffic management, though it may be more limited compared to Istio® or Cilium® regarding granular control.	Supports extensive customization via dynamic routing configurations and custom logic, although specific implementations may require additional effort.

Based on this comparison, among open-source solution candidates Cilium®, Istio®, Linkerd, and Envoy, no single tool stands out as the absolute best; instead, the choice depends on specific use cases and functionality requirements.

Based on the overall analysis in clauses A.1.4, A.2.4, and A.4.3, most Traffic Enforcer functional and interface requirements can be met by the considered tools. However, Cilium® and Istio® stand out as the most suitable options. The choice ultimately depends on the specific use cases and functionalities required. For example, Cilium® excels at layer 4 networking, leveraging eBPF technology for efficient traffic management with minimal overhead, particularly in cloud-native environments. This makes it ideal for scenarios requiring high-performance, low-latency traffic control. In contrast, Istio® is more focused on layer 7 traffic management, offering advanced features such as sophisticated routing, policy enforcement, and observability. This makes Istio® preferable for complex service mesh scenarios that require detailed control and monitoring of microservice communications.

A.4.4 Cross-comparison of open-source solutions for Notification Manager Function

Based on the solutions considered in clause A.1.8, Table A.4.4-1 shows a cross-comparison of the open-source solutions (Prometheus Alertmanager, Argo®, Kafka and Sensu) based on defined criteria focusing on functional, interface, and non-functional requirements.

Table A.4.4-1: Cross-comparison of open-source solutions for Notification Manager Function

Cross comparison criteria	Prometheus Alertmanager	Argo®	Kafka	Sensu
NotificationManager.001	Prometheus Alertmanager can receive alerts only from Prometheus-compatible monitoring systems using HTTP API or Alertmanager federation. However, it lacks built-in support for direct VNF or PaaS service notifications unless those services integrate with Prometheus exporters.	Does not have a native mechanism to receive notifications from external services.	Kafka is a distributed event streaming platform that natively supports receiving notifications from VNFs & PaaS services. Notifications can be sent to Kafka via APIs, message queues, or connectors, ensuring high availability, scalability, and real-time processing.	Sensu can receive monitoring events and notifications from VNFs & PaaS services via API, agents, and service hooks. However, it is less scalable than Kafka for large-scale distributed environments.

Cross comparison criteria	Prometheus Alertmanager	Argo®	Kafka	Sensu
NotificationManager.002	Prometheus Alertmanager provides built-in deduplication, grouping, and inhibition of alerts. It reduces duplicate alerts and sends grouped notifications based on labels and routing configurations. However, it cannot perform advanced event correlation beyond basic matching rules.	Does not have any built-in capability to process, group, or deduplicate notifications.	Does not process notifications itself, but allows event processing frameworks like Kafka Streams and kSQL which enable grouping, filtering, deduplication, and correlation of notifications in real time.	Sensu provides event filtering and deduplication to reduce alert noise. However, its deduplication logic is limited to event occurrences and lacks advanced event correlation and transformation capabilities like Kafka Streams.
NotificationManager.003	Alertmanager routes alerts to predefined receivers (email, webhooks, etc.) based on label-based routing rules. However, it lacks dynamic consumer authorization controls (no fine-grained access management).	It is a workflow orchestration tool, It does not handle notification routing.	Kafka natively supports event routing to multiple authorized consumers: • Consumer groups allow multiple services to receive notifications. • ACLs & RBAC provide fine-grained access control. • Kafka Streams & kSQL allow filtering and transformation of messages before routing.	Sensu supports event routing to specific handlers (e.g. webhooks, alert services). It provides basic access control via RBAC, but lacks Kafka's scalability and dynamic subscription management.
NotificationManager.004	Alertmanager does not provide dynamic subscription management. Routing is statically defined in configuration files, and consumers cannot manage their subscriptions dynamically.	Does not support event-driven subscription management.	Subscription management needs to be handled by external authentication mechanisms (e.g. Kafka ACLs, OAuth, or RBAC in applications). Kafka natively supports consumer-managed subscriptions via consumer groups, topic-based pub/sub model, and access control mechanisms. Consumers can dynamically subscribe/unsubscribe to specific event streams.	Sensu supports basic subscription management via event handlers and RBAC. However, it lacks Kafka's dynamic topic-based subscription model and scalability.

Cross comparison criteria	Prometheus Alertmanager	Argo®	Kafka	Sensu
NotificationManagerInf.001	Alertmanager produces notifications via its API and routes alerts to configured receivers (email, webhooks, etc.). However, it lacks a dedicated interface for managing notifications dynamically (everything is configured statically via YAML).	It is a workflow automation tool, not a notification manager. It does not produce or expose a notifications management interface.	Kafka's API allows applications to produce and manage notifications dynamically, offering high throughput, persistence, and fine-grained access control.	Sensu produces notifications via event pipelines and handlers. It exposes an API to manage events, making it a viable notification management interface for monitoring systems. However, it lacks Kafka's high scalability and message persistence.
Notif.Manager Notif.Mgmt.001	It does not provide dynamic subscription management. Instead, notifications are routed based on predefined static configurations (YAML-based routing rules). Consumers cannot dynamically subscribe/unsubscribe	Argo® is a workflow automation tool, not designed for notification handling or subscription management.	It does not offer a built-in API for dynamically managing consumer subscriptions but supports a publish-subscribe model, allowing consumers to subscribe or unsubscribe from topics dynamically. Additionally, it provides fine-grained access control for subscriptions through ACLs and RBAC.	Supports subscription-based event routing via configuration and API, with predefined subscriptions rather than dynamic runtime management.
Notif.Manager Notif.Mgmt.002	Prometheus Alertmanager is useful for alert-based notifications but lacks access control & persistence.	Argo® is a workflow automation tool. It can trigger workflows but does not natively handle notifications to external consumers.	Provides the most robust notification system. It ensures reliable delivery, persistence, and consumer-based access control.	Enables the delivery of processed notifications to authorize consumers via handlers, RBAC, and subscriptions. Notifications are processed in real-time but are not retained for future retrieval.
CRD Support	Provides CRD for managing Alertmanager instances but lacks built-in dynamic subscription handling.	Offers CRD support via Kubernetes® for workflows and event-driven automation, but not specifically for notifications.	Does not natively support CRD but integrates with Strimzi or Confluent Operator, CRD-based management of Kafka topics and consumer groups.	Supports CRD via Sensu Go API with custom event processing and routing configurations.
Community and Ecosystem	Maintains an active and well-established community with ongoing development and contributions.	Growing ecosystem with strong Kubernetes®-native integration and increasing adoption.	Large, mature community with extensive enterprise adoption and ecosystem of plugins.	Active open-source community with contributions focused on observability and monitoring
Scalability	Supports horizontal scaling via sharding and clustering but requires additional tools like Thanos for large-scale deployments.	Designed for Kubernetes-native scalability, leveraging Kubernetes® controllers for efficient workload management.	Highly scalable, built for distributed message streaming, capable of handling large-scale event processing.	Supports distributed architectures but scales primarily through agent-based workload distribution.

Cross comparison criteria	Prometheus Alertmanager	Argo®	Kafka	Sensu
Customization and Extensibility	Offers Alert templates & notification routing rules. Relies on external integrations (Grafana®, Webhooks, or custom scripts).	Uses Argo® Workflows & Argo® Events for automation scripting and supporting YAML-based workflow customization.	Allows custom event processing logic using Kafka Streams API (Java/Scala).	Supports scripting (Go, Python, Ruby) for event handling.

Based on this comparison, among the open-source solution candidates Prometheus Alertmanager, Argo®, Kafka, and Sensu, no single tool stands out as the absolute best; instead, the choice depends on specific use cases and functionality requirements.

Based on the overall analysis in clauses A.1.8, A.2.5 and A.4.4, most Notification Manager functional and interface requirements can be met by the considered open-source tools. However, Kafka and Prometheus Alertmanager stand out as the most suitable options. The choice ultimately depends on the specific use cases and functionalities required. For example, Kafka is ideal for high-volume, scalable, and event-driven architectures, providing dynamic consumer management, fine-grained access control. It ensures efficient routing of notifications to authorized consumers and supports strong access control. On the other hand, Prometheus Alertmanager is optimized for monitoring alerts. It is best suited for alerting and monitoring use cases where notifications follow predefined static routes based on labels.

A.4.5 Cross-comparison of open-source solutions for Metrics Analyser Function

Based on the tools considered in clause A.1.7, Table A.4.5-1 shows a cross-comparison of the open-source solutions (Coroot, OpenSearch, and VictoriaMetrics) based on defined criteria focused on functional, interface, and non-functional requirements.

Table A.4.5-1: Cross-comparison of open-source solutions for Metrics Analyser Function

Cross comparison criteria	Coroot	OpenSearch	VictoriaMetrics
VNFMetricAnalyser.001	Provides built-in analysis for performance issues and application-level anomalies using Prometheus data.	Supports advanced metrics analysis with custom queries, aggregations, and anomaly detection through plugins.	Supports time-series analysis with MetricsQL, but advanced analytics like anomaly detection require external customization.
VNFMetricAnalyser.002	Provides basic configuration options for adjusting thresholds and defining SLOs for commonly monitored metrics.	Fully configurable via dashboards, queries, and alerting rules, supporting custom analysis pipelines.	Supports static rule-based configurations via VMRule but lacks dynamic or runtime configurability.
VNFMetricAnalyser.003	Provides basic webhook notifications for incidents, lacking advanced alert routing features.	Provides built-in alerting with routing to multiple channels including email and webhooks.	Provides advanced alert routing, grouping, and suppression with external notification support.
VNFMetricAnalyser.004	Provides web dashboards for sharing analytics with authorized users.	Supports secure, role-based access to dashboards and analytics with fine-grained user permissions.	Exposes results via Query API with basic security, but advanced access control requires external integration.
VNFMetricAnalyser.001	Provides observability dashboards and analysis using eBPF and Prometheus data, covering metrics, logs, traces, and profiles.	Provides a complete analysis and visualization pipeline with integration for Prometheus and OpenTelemetry in OpenSearch Dashboards.	Provides an Query API for exposing analytics results such as recording rule outputs and alert statuses.

Cross comparison criteria	Coroot	OpenSearch	VictoriaMetrics
MetricAnalyser.Expose.001	Supports result sharing via dashboards and notifications with basic RBAC, though access control may be limited in some setups.	Provides secure, role-based access to metrics dashboards, ensuring controlled exposure of analysis results to authorized users.	Exposes analytics through Query APIs but requires external components for enforcing authentication and authorization.
MetricAnalyser.Expose.002	Allows threshold and SLO configuration for metrics processing but lacks flexibility beyond predefined parameters.	Provides advanced, customizable metric processing through queries, pipelines, and transformation rules for real-time analysis.	Supports static rule-based configuration using VMRule but lacks dynamic pipeline management or advanced processing control.
CRD Support	Offers a Kubernetes® Operator that simplifies deployment and management within Kubernetes® clusters. While it utilizes Helm charts, it doesn't define extensive CRDs for granular configurations.	Provides comprehensive CRDs support, allowing detailed management of OpenSearch clusters and dashboards through custom resources.	Provides multiple CRDs (e.g. VMCluster, VMRule, VMAgent), facilitating detailed configuration and management of its components within Kubernetes® environments.
Community and Ecosystem	Has an active community with resources like Slack channels, GitHub repositories, and documentation. Its ecosystem is growing, focusing on observability and performance monitoring.	Has a vibrant community-driven ecosystem, with extensive documentation, forums, and a wide range of plugins and integrations, based on the foundation of Elasticsearch.	Has a steadily growing community, offering comprehensive documentation, GitHub repositories, and integrations with tools like Grafana® and Prometheus.
Scalability	Designed for simplicity, scales effectively for small to medium-sized deployments. Its architecture supports horizontal scaling but may require additional configurations for large-scale environments.	Supports both vertical and horizontal scaling. It can handle large volumes of data and complex queries, making it suitable for enterprise-level deployments.	Designed for high performance and scalability, it can handle millions of data points per second. It works in both single-node and cluster setups, making it efficient for storing and accessing time-series data.
Customization and Extensibility	Provides customization options through its configuration files and supports integrations with various data sources. However, its extensibility is somewhat limited compared to more mature platforms.	Offers extensive customization and extensibility through its plugin architecture, allowing users to develop custom plugins and leverage existing ones to enhance functionality.	Supports customization via its MetricsQL query language and configuration files. While it offers essential extensibility features, advanced customizations may require additional tooling or integrations.

Based on this comparison, among open-source solution candidates Coroot, OpenSearch, and VictoriaMetrics, no single tool stands out as the absolute best; instead, the choice depends on specific use cases and functionality requirements.

Based on the overall analysis in clauses A.1.7, A.2.7, and A.4.5, most Metrics Analyser functional and interface requirements can be met by the considered tools. However, VictoriaMetrics and OpenSearch stand out as the most suitable options. The choice ultimately depends on the specific use cases and functionalities required. For example, VictoriaMetrics is best suited for handling large amounts of time-series data, offering fast processing, efficient storage, and quick querying, making it ideal for heavy environments. In contrast, OpenSearch provides a unified observability platform with native RBAC, dashboards, and ML-driven anomaly detection, optimized for interactive analysis.

A.5 Example CRD schemas

A.5.1 OpenSearch resource

A.5.1.1 OpenSearchCluster

This example includes a subset of the OpenSearchCluster CRD [13] that has been considered when performing the input parameter mapping in clause 8.6.1.

```
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  annotations:
    controller-gen.kubebuilder.io/version: v0.14.0
  name: opensearchclusters.opensearch.opster.io
spec:
  group: opensearch.opster.io
  names:
    kind: OpenSearchCluster
    listKind: OpenSearchClusterList
    plural: opensearchclusters
    singular: opensearchcluster
  scope: Namespaced
  versions:
    - name: v1
      schema:
        openAPIV3Schema:
          description: "Schema for OpenSearchCluster API"
          type: object
          properties:
            apiVersion: {type: string, description: "Defines the versioned schema of
the object."}
            kind: {type: string, description: "Represents the REST resource this object
corresponds to."}
            metadata: {type: object}
            spec:
              description: ClusterSpec defines the desired state of OpenSearchCluster
              type: object
              required: [nodePools]
              properties:
                general:
                  description: INSERT ADDITIONAL SPEC FIELDS - desired state of cluster
                  type: object
                  properties: {serviceName: {type: string}, version: {type: string}}
                  required: [serviceName]
                dashboards:
                  description: Dashboards configuration.
                  type: object
                  properties:
                    enable: {type: boolean}
                    replicas: {type: integer, format: int32}
                    version: {type: string}
                    resources:
                      description:  ResourceRequirements describes the compute resource
requirements.
                      type: object
                      properties:
                        limits:
```

```
                        description: Limits describes the maximum amount of compute
resources allowed.
                        additionalProperties:
                          anyOf: [type: integer, type: string]
                          pattern: ^(\+|-)?(([0-9]+(\.[0-9]*)?)|(\.[0-
9]+))(([KMGTPE]i)|[numkMGTPE]|([eE](\+|-)?(([0-9]+(\.[0-9]*)?)|(\.[0-9]+))))?$
                          x-kubernetes-int-or-string: true
                          type: object
                      requests:
                        description: Requests describes the minimum amount of compute
resources required. If Requests is omitted for a container, it defaults to Limits if
that is explicitly specified, otherwise to an implementation-defined value. Requests
cannot exceed Limits.
                        additionalProperties:
                          anyOf: [type: integer, type: string]
                          pattern: ^(\+|-)?(([0-9]+(\.[0-9]*)?)|(\.[0-
9]+))(([KMGTPE]i)|[numkMGTPE]|([eE](\+|-)?(([0-9]+(\.[0-9]*)?)|(\.[0-9]+))))?$
                          x-kubernetes-int-or-string: true
                          type: object
                required: [replicas, version]
              nodePools:
                description: "Node pools configuration"
                type: array
                items:
                  type: object
                  properties:
                    component: {type: string}
                    replicas: {type: integer, format: int32}
                    diskSize: {type: string}
                    nodeSelector: {additionalProperties: {type: string}, type:
object}
                    resources:
                      description: Resource requirements describes the compute
resource requirements.
                      type: object
                      properties:
                        limits:
                          description: Limits describes the maximum amount of compute
resources allowed.
                          additionalProperties:
                            anyOf: [type: integer, type: string]
                            pattern: ^(\+|-)?(([0-9]+(\.[0-9]*)?)|(\.[0-
9]+))(([KMGTPE]i)|[numkMGTPE]|([eE](\+|-)?(([0-9]+(\.[0-9]*)?)|(\.[0-9]+))))?$
                            x-kubernetes-int-or-string: true
                            type: object
                        requests:
                            description: Requests describes the minimum amount of
compute resources required. If Requests is omitted for a container, it defaults to
Limits if that is explicitly specified, otherwise to an implementation-defined value.
Requests cannot exceed Limits.
                          additionalProperties:
                            anyOf: [type: integer, type: string]
                            pattern: ^(\+|-)?(([0-9]+(\.[0-9]*)?)|(\.[0-
9]+))(([KMGTPE]i)|[numkMGTPE]|([eE](\+|-)?(([0-9]+(\.[0-9]*)?)|(\.[0-9]+))))?$
                            x-kubernetes-int-or-string: true
                            type: object
                    roles: {type: array, items: {type: string}}
                  required: [component, replicas, roles]
```

A.5.2 Fluent Bit resources

A.5.2.1 ClusterFilter

This example includes a subset of the ClusterFilter CRD [12] that has been considered when performing the input parameter mapping in clause 7.6.1.

```
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  annotations:
    controller-gen.kubebuilder.io/version: v0.15.0
  name: clusterfilters.fluentbit.fluent.io
spec:
  group: fluentbit.fluent.io
  names:
    kind: ClusterFilter
    listKind: ClusterFilterList
    plural: clusterfilters
    singular: clusterfilter
  scope: Cluster
  versions:
    - name: v1alpha2
      schema:
        openAPIV3Schema:
          description: ClusterFilter defines a cluster-level Filter configuration.
          type: object
          properties:
            apiVersion: {description: API version of the resource., type: string}
            kind: {description: Kind of the resource., type: string}
            metadata: {type: object}
            spec:
              type: object
              description: Specification of desired Filter configuration.
              properties:
                filters:
                  type: array
                  description: A set of filter plugins in order.
                  items:
                    type: object
                    properties:
                      grep:
                        type: object
                        description: Grep defines Grep Filter configuration.
                        properties:
                          exclude: {type: string, description: "Exclude records which
field matches the regular expression. Value Format: FIELD REGEX."}
                          regex: {type: string, description: "Keep records which field
matches the regular expression. Value Format: FIELD REGEX."}
                      kubernetes:
                        type: object
                        description: Kubernetes defines Kubernetes Filter
configuration.
                        properties:
                          kubeURL: {type: string, description: API Server endpoint.}
                          labels: {type: boolean, description: Include Kubernetes
resources labels in the extra metadata.}
                          regexParser: {type: string, description: Set an alternative
Parser to process record Tag and extract pod_name, namespace_name, container_name and
```

```
docker_id. The parser must be registered in a parsers file (refer to parser filter-
kube-test as an example).}
                        mergeLog: {type: boolean, description: When enabled, it
checks if the log field content is a JSON string map, if so, it append the map fields
as part of the log structure.}
                        annotations: {type: boolean, description: Include Kubernetes
resource annotations in the extra metadata.}
                    modify:
                      type: object
                      description: Modify defines Modify Filter configuration.
                      properties:
                        alias: {type: string, description: Alias for the plugin.}
                        conditions:
                          type: array
                          description: All conditions have to be true for the rules
to be applied.
                          items:
                            type: object
                            properties:
                              aKeyMatches: {type: string, description: Is true if a
key matches regex KEY.}
                              keyDoesNotExist: {type: object, description: Is true if
KEY does not exist, additionalProperties: {type: string}}
                              keyValueDoesNotMatch: {type: object, description: Is
true if key KEY exists and its value does not match VALUE, additionalProperties: {type:
string}}
                        rules:
                          type: array
                          description: Rules are applied in the order they appear,
with each rule operating on the result of the previous rule.
                          items:
                            type: object
                            properties:
                              add: {type: object, description: Add a key/value pair
with key KEY and value VALUE if KEY does not exist, additionalProperties: {type:
string}}
                              remove: {type: string, description: Remove a key/value
pair with key KEY if it exists }
                              rename: {type: object, description: Rename a key/value
pair with key KEY to RENAMED_KEY if KEY exists AND RENAMED_KEY does not exist,
additionalProperties: {type: string}}
                    parser:
                      type: object
                      description: Parser defines Parser Filter configuration.
                      properties:
                        keyName: {type: string, description: Specify field name in
record to parse.}
                        parser: {type: string, description: Specify the parser name
to interpret the field. Multiple Parser entries are allowed (split by comma).}
                        preserveKey: {type: boolean, description: Keep original
Key_Name field in the parsed result. If false, the field will be removed.}
                        reserveData: {type: boolean, description: Keep all other
original fields in the parsed result. If false, all other original fields will be
removed.}
                        unescapeKey: {type: boolean, description: 'If the key is a
escaped string (e.g: stringify JSON), unescape the string before to apply the parser.'}
```

A.5.2.2 ClusterOutput

This example includes a subset of the ClusterOutput CRD [12] that has been considered when performing the input parameter mapping in clause 7.6.1.

```
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  annotations:
    controller-gen.kubebuilder.io/version: v0.15.0
  name: clusteroutputs.fluentbit.fluent.io
spec:
  group: fluentbit.fluent.io
  names:
    kind: ClusterOutput
    listKind: ClusterOutputList
    plural: clusteroutputs
    singular: clusteroutput
  scope: Cluster
  versions:
    - name: v1alpha2
      schema:
        openAPIV3Schema:
          description: ClusterOutput is the Schema for the cluster-level outputs API
          type: object
          properties:
            apiVersion: {description: Defines the versioned schema of this object.,
type: string}
            kind: {description: Represents the REST resource type in CamelCase., type:
string}
            metadata: {type: object}
            spec:
              description: OutputSepc defines the desired state of ClusterOutput
              type: object
              properties:
                file:
                  description: File Output configuration
                  type: object
                  properties:
                    delimiter: {description: The character to separate each pair.
Applicable only if format is csv or ltsv., type: string}
                    file: {description: Set file name to store the records. If not set,
the file name will be the tag associated with the records., type: string}
                    format:
                      description: 'The format of the file content.'
                      enum: [out_file, plain, csv, ltsv, template]
                      type: string
                    path: {description: Absolute directory path to store files. If not
set, Fluent Bit will write the files on its own positioned directory., type: string}
                opentelemetry:
                  description: OpenTelemetry defines OpenTelemetry Output
configuration.
                  type: object
                  properties:
                    addLabel: {description: This allows you to add custom labels to all
metrics exposed through the OpenTelemetry exporter. You may have multiple of these
fields., additionalProperties: {type: string}, type: object}
                    header: {description: Add a HTTP header key/value pair. Multiple
headers can be set., additionalProperties: {type: string}, type: object}
                    host: {description: IP address or hostname of the target HTTP
Server, default `127.0.0.1`, type: string}
```

```
                    networking:
                      description: Include fluentbit networking options for this
output-plugin.
                      type: object
                      properties:
                        DNSMode: {description: Select the primary DNS connection type
(TCP or UDP)., enum: [TCP, UDP], type: string}
                        DNSPreferIPv4: {description: Prioritize IPv4 DNS results when
trying to establish a connection., type: boolean}
                        sourceAddress: {description: Specify network address to bind
for data traffic., type: string}
                    port: {description: TCP port of the target OpenTelemetry instance,
default `80`.', format: int32, minimum: 1, maximum: 65535, type: integer}
```

A.5.3 Istio® resources

A.5.3.1 DestinationRule

This example includes a subset of the DestinationRule CRD [11] that has been considered when performing the input parameter mapping in clause 6.6.1.

```
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  name: destinationrules.networking.istio.io
spec:
  group: networking.istio.io
  names:
    kind: DestinationRule
    plural: destinationrules
    singular: destinationrule
  scope: Namespaced
  versions:
    - name: v1
      schema:
        openAPIV3Schema:
          type: object
          properties:
            spec:
              description: "Configuration affecting load balancing, outlier detection,
etc."
              type: object
              properties:
                host: {description: The name of a service from the service registry.,
type: string}
                subsets:
                  description: One or more named sets that represent individual
versions of a service.
                  type: array
                  items:
                    type: object
                    properties:
                      labels:
                        description: Labels apply a filter over the endpoints of a
service in the service registry.
                        type: object
                        additionalProperties: {type: string}
                      name: {description: Name of the subset., type: string}
```

```
                    trafficPolicy:
                      description: Traffic policies that apply to this subset.
                      type: object
                      properties:
                        loadBalancer:
                          description: Settings controlling the load balancer
algorithms.
                          type: object
                          properties:
                            simple: {type: string, enum: [UNSPECIFIED, LEAST_CONN,
RANDOM, PASSTHROUGH, ROUND_ROBIN, LEAST_REQUEST]}
                          oneOf:
                            - not:
                                anyOf:
                                  - required: [simple]
                                  - required: [consistentHash]
                            - required: [simple]
                            - required: [consistentHash]
                  required: [name]
```

A.5.3.2 AuthorizationPolicy

This example includes a subset of the AuthorizationPolicy CRD [11] that has been considered when performing the input parameter mapping in clause 6.6.1.

```
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  name: authorizationpolicies.security.istio.io
spec:
  group: security.istio.io
  names:
    kind: AuthorizationPolicy
    plural: authorizationpolicies
    singular: authorizationpolicy
  scope: Namespaced
  versions:
    - name: v1
      schema:
        openAPIV3Schema:
          properties:
            spec:
              description: Configuration for access control on workloads.
              properties:
                action: { description: Optional, enum: [ALLOW, DENY, AUDIT, CUSTOM],
type: string}
                rules:
                  description: Optional.
                  items:
                    type: object
                    properties:
                      from:
                        description: Optional.
                        items:
                          type: object
                          properties:
                            source:
                              description: Source specifies the source of a request.
                              type: object
```

```
                            properties:
                              ipBlocks: {description: Optional., items: {type:
string}, type: array}
                              namespaces: {description: Optional., items: {type:
string}, type: array}
                              principals: {description: Optional., items: {type:
string}, type: array}
                        type: array
                      to:
                        description: Optional.
                        items:
                          properties:
                            operation:
                              description: Operation specifies the operation of a
request.
                              type: object
                              properties:
                                hosts: {description: Optional., items: {type: string},
type: array}
                                methods: {description: Optional., items: {type:
string}, type: array}
                                paths: {description: Optional., items: {type: string},
type: array}
                                ports: {description: Optional., items: {type: string},
type: array}
                          type: object
                        type: array
                      when:
                        description: Optional.
                        items:
                          type: object
                          properties:
                            key: {description: The name of an Istio attribute., type:
string}
                            notValues: {description: Optional., items: {type: string},
type: array}
                            values: {description: Optional., items: {type: string},
type: array}
                          required: [key]
                        type: array
                  type: array
                selector:
                  properties:
                    matchLabels:
                      description: One or more labels that indicate a specific set of
pods/VMs on which a policy should be applied.
                      type: object
                      additionalProperties:
                        type: string
                  type: object
              type: object
          type: object
```

A.5.4 Prometheus Alertmanager resources

A.5.4.1 AlertmanagerConfig

This example includes a subset of the AlertmanagerConfig CRD [14] that has been considered when performing the input parameter mapping in clause 9.6.1.

```
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  name: alertmanagerconfigs.monitoring.coreos.com
spec:
  group: monitoring.coreos.com
  names:
    plural: alertmanagerconfigs
    singular: alertmanagerconfig
    kind: AlertmanagerConfig
    listKind: AlertmanagerConfigList
  scope: Namespaced
  versions:
    - name: v1alpha1
      served: true
      storage: true
      schema:
        openAPIV3Schema:
          description: AlertmanagerConfig configures the Prometheus Alertmanager,
specifying how alerts should be grouped, inhibited and notified to external systems.
          type: object
          required: [spec]
          properties:
            metadata: {type: object}
            spec:
              type: object
              description: AlertmanagerConfigSpec is a specification of the desired
behavior of the Alertmanager configuration.
              properties:
                route:
                  type: object
                  description: "The Alertmanager route definition for alerts matching
the resource's namespace. If present, it will be added to the generated Alertmanager
configuration as a first-level route."
                  properties:
                    receiver: {type: string, description: Name of the receiver for this
route. If not empty, it should be listed in the `receivers` field.}
                    matchers:
                      description: "List of matchers that the alert's labels should
match. For the first level route, the operator removes any existing equality and regexp
matcher on the `namespace` label and adds a `namespace: <objectnamespace>` matcher".
                      type: array
                      items:
                        type: object
                        properties:
                          name: {description: Label to match., type: string}
                          value: {description: Label value to match., type: string}
                          matchType: {description: Match operation available with
AlertManager >= v0.22.0 and takes precedence over Regex (deprecated) if non-empty.,
type: string, enum: ["=", "!="]}
                          regex: {description: "Whether to match on equality (false) or
regular-expression (true). Deprecated: for AlertManager >= v0.22.0, `matchType` should
be used instead.",type: Boolean}
```

```
                      required: [name]
                  groupBy:
                    type: array
                    description: List of labels to group by. Labels must not be
repeated (unique list). Special label "..." (aggregate by all possible labels), if
provided, must be the only element in the list.
                    items: {type: String}
                receivers:
                  type: array
                  description: List of receivers.
                  items:
                    type: object
                    properties:
                      name: {type: string, description: Name of the receiver. Must be
unique across all items from the list., minLength: 1}
                      emailConfigs:
                        type: array
                        description: List of Email configurations.
                        items:
                          type: object
                          properties:
                            to: {type: string, description: The email address to send
notifications to.}
                            from: {type: string, description: "The sender address."}
                            smarthost: {type: string, description: The SMTP host and
port through which emails are sent. E.g. example.com:25}
                            authUsername: {type: string, description: The username to
use for authentication.}
                        required: [name]
                inhibitRules:
                  type: array
                  description: List of inhibition rules. The rules will only apply to
alerts matching the resource's namespace.
                  items:
                    type: object
                    properties:
                      sourceMatch:
                        type: array
                        description: Matchers for which one or more alerts have to
exist for the inhibition to take effect. The operator enforces that the alert matches
the resource's namespace.
                        items:
                          type: object
                          properties:
                            matchType: {description: "Match operation available with
AlertManager >= v0.22.0 and takes precedence over Regex (deprecated) if non-empty.",
enum: ['!=', '=', '=~', '!~'], type: string}
                            name: {description: Label to match., minLength: 1, type:
string}
                            regex: {description: "Whether to match on equality (false)
or regular-expression (true). Deprecated: for AlertManager >= v0.22.0, `matchType`
should be used instead.", type: Boolean}
                          required: [name]
                      targetMatch:
                        type: array
                        description: Matchers that have to be fulfilled in the alerts
to be muted. The operator enforces that the alert matches the resource's namespace.
                        items:
                          type: object
                          properties:
```

```
                            matchType: {description: "Match operation available with
AlertManager >= v0.22.0 and takes precedence over Regex (deprecated) if non-empty.",
enum: ['!=', '=', '=~', '!~'], type: string}
                            name: {description: Label to match., minLength: 1, type:
string}
                            regex: {description: "Whether to match on equality (false)
or regular-expression (true). Deprecated: for AlertManager >= v0.22.0, `matchType`
should be used instead.", type: Boolean}
                          required: [name]
                        equal:
                          type: array
                          items:
                            type: string
                            description: Labels that must have an equal value in the
source and target alert for the inhibition to take effect.
```

A.5.4.2 Alertmanager

This example includes a subset of the Alertmanaer CRD [14] that has been considered when performing the input parameter mapping in clause 9.6.1.

```
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  name: alertmanagers.monitoring.coreos.com
spec:
  group: monitoring.coreos.com
  names:
    plural: alertmanagers
    singular: alertmanager
    kind: Alertmanager
    listKind: AlertmanagerList
  scope: Namespaced
  versions:
    - name: v1
      served: true
      storage: true
      schema:
        openAPIV3Schema:
          type: object
          description: The Alertmanager CRD defines a desired Alertmanager setup to run
in a Kubernetes cluster. It allows to specify many options such as the number of
replicas, persistent storage and many more.
          Required: [spec]
          properties:
            metadata: {type: object}
            spec:
              type: object
              description: Specification of the desired behavior of the Alertmanager
cluster.
              properties:
                storage:
                  type: object
                  description: Storage is the definition of how storage will be used by
the Alertmanager instances.
                  properties:
                    volumeClaimTemplate:
                      type: object
```

```
                    description: "Defines the PVC spec to be used by the Prometheus
StatefulSets. The easiest way to use a volume that cannot be automatically provisioned
is to use a label selector alongside manually created PersistentVolumes."
                    properties:
                      metadata: {type: object, description: EmbeddedMetadata contains
metadata relevant to an EmbeddedResource., properties: {name: {type: string,
description: Name must be unique within a namespace. Is required when creating
resources, although some resources may allow a client to request the generation of an
appropriate name automatically. Name is primarily intended for creation idempotence and
configuration definition.}}}
                      spec: {type: object, description: description: "Defines the
desired characteristics of a volume requested by a pod author.", properties:
{volumeMode: {type: string, description: "volumeMode defines what type of volume is
required by the claim. Value of Filesystem is implied when not included in claim
spec."}, volumeName: {type: string, description: "volumeName is the binding reference
to the PersistentVolume backing this claim.}}}
                alertmanagerConfigSelector:
                  type: object
                  description: AlertmanagerConfigs to be selected for to merge and
configure Alertmanager with.
                  x-kubernetes-map-type: atomic
                  properties:
                    matchExpressions:
                      type: object
                      description: matchExpressions is a list of label selector
requirements. The requirements are ANDed.
                      items:
                        type: array
                        x-kubernetes-list-type: atomic
                        description: "A label selector requirement is a selector that
contains values, a key, and an operator that relates the key and values."
                        properties:
                          key: {type: string, description: key is the label key that
the selector applies to.}
                          operator: {type: string, description: operator represents a
key's relationship to a set of values. Valid operators are In, NotIn, Exists and
DoesNotExist.}
                      required: [key, operator]
```

A.5.5 VictoriaMetrics resource

A.5.5.1 VMRule

This example includes a subset of the VMRule CRD [9] that has been considered when performing the input parameter mapping in clause 10.6.1.

```
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  annotations:
    controller-gen.kubebuilder.io/version: v0.17.2
  name: vmrules.operator.victoriametrics.com
spec:
  group: operator.victoriametrics.com
  names:
    kind: VMRule
    listKind: VMRuleList
    plural: vmrules
```

```
    singular: vmrule
  scope: Namespaced
  versions:
  - name: v1beta1
    schema:
      openAPIV3Schema:
          type: object
          description: VMRule defines rule records for vmalert application.
          required: [spec]
          properties:
            apiVersion: {description: API version of the VMRule., type: string}
            kind: {description: Kind of the resource., type: string}
            metadata: {type: object}
            spec:
              type: object
              description: VMRuleSpec defines the desired state of VMRule.
              required: [groups]
              properties:
                groups:
                  type: array
                  description: Groups list of group rules.
                  items:
                    type: object
                    required: [name, rules]
                    properties:
                      name: {type: string, description: Name of the rule group.}
                      interval: {type: string, description: Evaluation interval for
group.}
                      rules:
                        type: array
                        description: Rule list of alert rules.
                        items:
                          type: object
                          description: Rule describes an alerting or recording rule.

                          properties:
                            expr: {type: string, description: Expr is query, that will
be evaluated at dataSource.}
                            alert: {type: string, description: Alert is a name for
alert.}
                            for: {type: string, description: For evaluation interval in
time. Duration format 30s, 1m, 1h or nanoseconds.}
                            labels:
                              type: object
                              additionalProperties: {type: string}
                              description: Labels will be added to rule configuration.
                            record: {type: string, description: Record represents a
query, that will be recorded to dataSource.}
```

Annex B (informative): Sequence diagrams

B.1 Sequence diagrams for the Traffic Enforcer profiled solution

B.1.1 Flow of creating AuthorizationPolicy and DestinationRule as a Traffic Management related NFV objects to manage traffic

This clause describes a sequence for creating an individual AuthorizationPolicy and DestinationRule as a Traffic Management related NFV objects to manage the traffic.

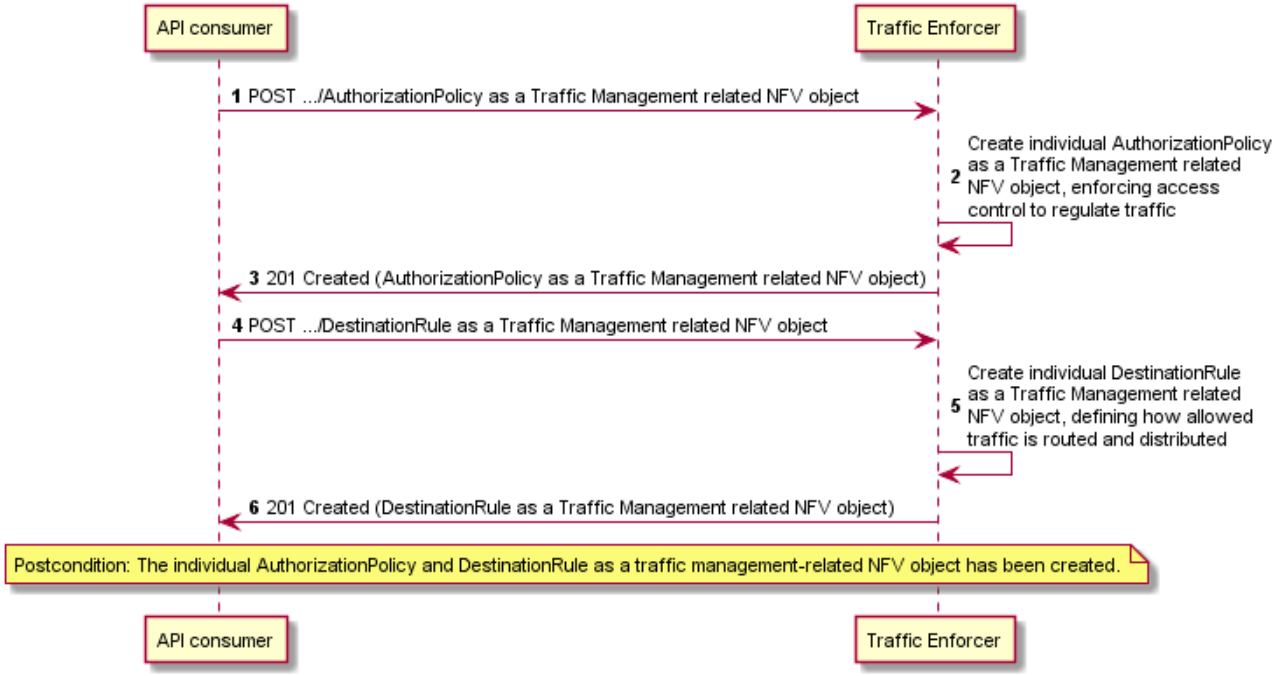

Figure B.1.1-1: Flow of creating AuthorizationPolicy and DestinationRule as a Traffic Management related NFV object

The creation of AuthorizationPolicy and DestinationRule as a traffic management related NFV objects, as illustrated in Figure B.1.1-1, consists of the following steps:

Precondition: none.

1) The API consumer sends a POST request to the Traffic Enforcer with the appropriate AuthorizationPolicy resource URI, including the data structure of the declarative descriptor of the respective AuthorizationPolicy custom resource object in the payload body, specifying traffic control rules, for example, based on source identity, namespaces, pod labels, hosts, ports, paths, and HTTP methods.

2) The Traffic Enforcer creates an individual AuthorizationPolicy as a Traffic Management related NFV object, enforcing access control to regulate traffic before it reaches the intended service or workload instances.

3) The Traffic Enforcer returns a "201 Created" response to the API consumer and includes in the payload body a representation of the created AuthorizationPolicy as a Traffic Management related NFV object.

4) The API consumer sends a POST request to the Traffic Enforcer with the appropriate DestinationRule resource URI, including the data structure of the declarative descriptor of the respective DestinationRule custom resource object in the payload body, specifying how allowed traffic should be further routed, for example, by defining service subsets, load balancing strategies, and connection policies.

5) The Traffic Enforcer creates an individual DestinationRule as a Traffic Management related NFV object, defining how allowed traffic is routed and distributed across service subsets.

6) The Traffic Enforcer returns a "201 Created" response to the API consumer and includes in the payload body a representation of the created DestinationRule as a Traffic Management related NFV object.

Postcondition: Upon successful completion, the individual AuthorizationPolicy and DestinationRule as the requested Traffic Management-related NFV objects has been created.

Error handling: In case of failure, appropriate error information is provided in the response.

B.2 Sequence diagram for the Log Aggregator profiled solution

B.2.1 Flow of log aggregation

This clause describes a sequence for log aggregation by creating individual ClusterFilter and ClusterOutput resources as a Log Exposure related NFV objects.

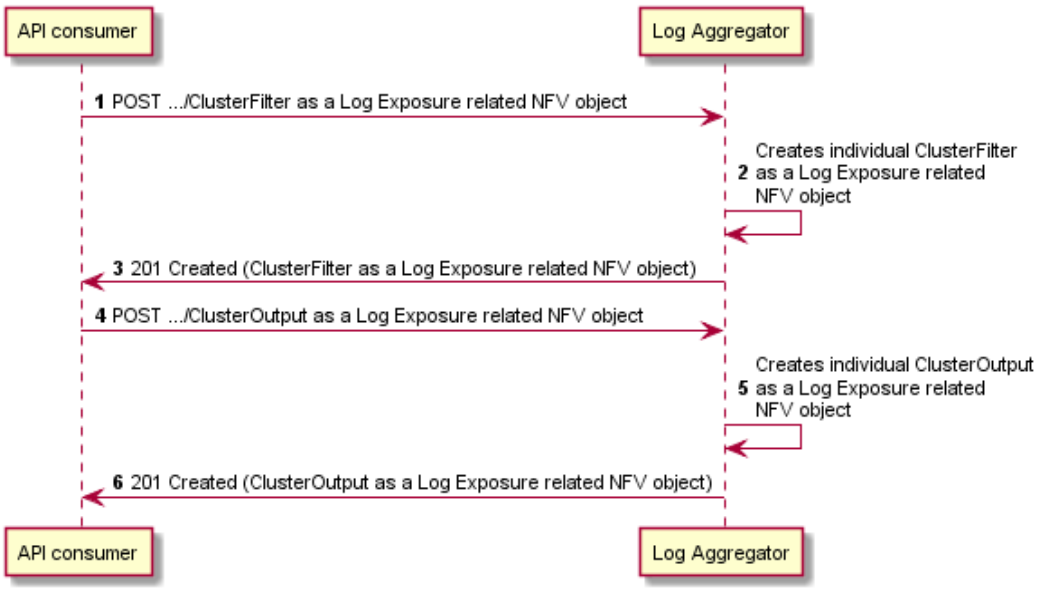

Figure B.2.1-1: Flow of log aggregation by creating an individual ClusterFilter and ClusterOutput resources as a Log Analysis Exposure related NFV object

The flow of log aggregation consists of the following steps as illustrated in Figure B.2.1-1:

Precondition: none.

1) The API consumer sends a POST request to the Log Aggregator with the appropriate ClusterFilter resource URI, including the data structure of the declarative descriptor of the respective ClusterFilter custom resource object in the payload body.

2) The Log Aggregator creates an individual ClusterFilter as a Log Exposure related NFV object to filter, modify, and structure logs based on the provided filter rules.

3) The Log Aggregator returns a "201 Created" response to the API consumer and includes in the payload body a representation of the created ClusterFilter as a Log Exposure related NFV object.

4) The API consumer sends a POST request to the Log Aggregator with the appropriate ClusterOutput resource URI, including the data structure of the declarative descriptor of the respective ClusterOutput custom resource object in the payload body.

5) The Log Aggregator creates an individual ClusterOutput as a Log Exposure related NFV object to configure the log destination based on the ClusterOutput configurations, ensuring log exposure for the analysis.

6) The Log Aggregator returns a "201 Created" response to the API consumer and includes in the payload body a representation of the created ClusterOutput as a Log Exposure related NFV object.

Postcondition: none.

Error handling: In case of failure, appropriate error information is provided in the response.

B.3 Sequence diagram for the Log Analyser profiled solution

B.3.1 Flow of log analysis

This clause describes a sequence for log analysis by creating an individual OpenSearchCluster resource as a Log Analysis Exposure related NFV object and executing a PPL query.

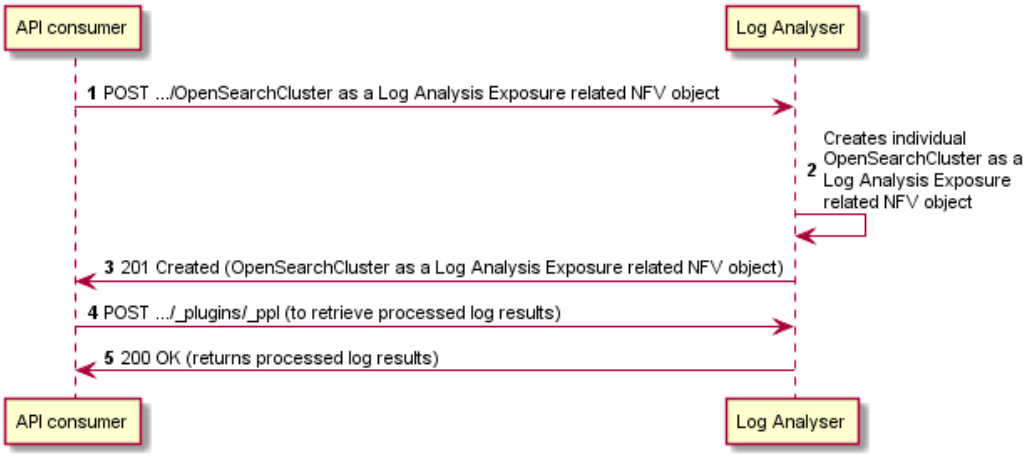

Figure B.3.1-1: Flow of log analysis by creating an individual OpenSearchCluster resource as a Log Analysis Exposure related NFV object and executing a PPL query

The flow of analysing logs consists of the following steps as illustrated in Figure B.3.1-1:

Precondition: none.

1) The API consumer sends a POST request to the Log Analyser with the appropriate OpenSearchCluster resource URI, including the data structure of the declarative descriptor of the respective OpenSearchCluster custom resource object in the payload body.

2) The Log Analyser creates an individual OpenSearchCluster as a Log Analysis Exposure related NFV object, to configure dashboard setup, log ingestion from source(s), and indexing and analysis settings.

3) The Log Analyser returns a "201 Created" response to the API consumer and includes in the payload body a representation of the created OpenSearchCluster as a Log Analysis Exposure related NFV object.

4) The API consumer sends a POST request to the PPL API endpoint (".../_plugins/_ppl") with a query string in the request body to retrieve the processed log results.

5) The Log Analyser returns a "200 OK" response to the API consumer, returning the processed logs in JDBC format, which is the default response structure for PPL.

Postcondition: None.

Error handling: In case of failure, appropriate error information is provided in the response.

B.4 Sequence diagrams for the Notification Manager profiled solution

B.4.1 Flow of creating AlertmanagerConfig and Alertmanager as a Notification Manager related NFV objects to manage notifications

This clause describes a sequence for creating an individual AlertmanagerConfig and Alertmanager as a Notification Manager related NFV objects to manage the notification.

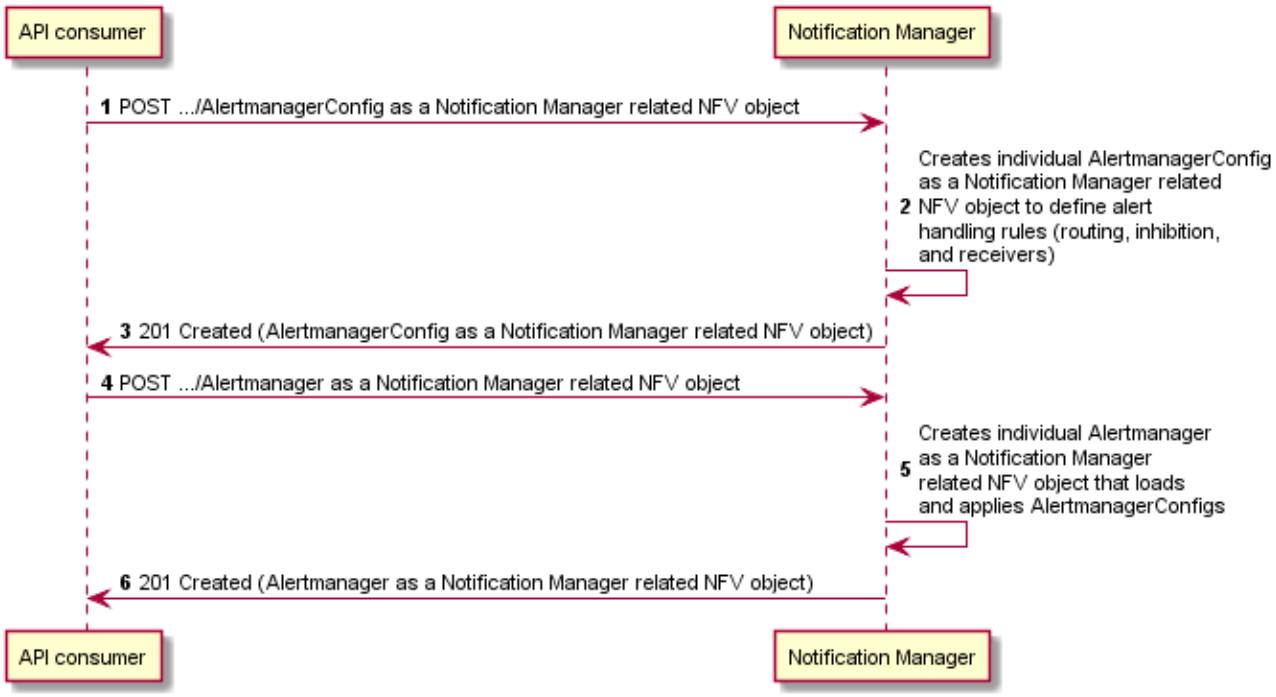

Figure B.4.1-1: Flow of creating AlertmanagerConfig and Alertmanager as a Notification Manager related NFV object

The creation of AlertmanagerConfig and Alertmanager as a notification manager related NFV objects, as illustrated in Figure B.4.1-1, consists of the following steps:

Precondition: none.

1) The API consumer sends a POST request to the Notification Manager with the appropriate AlertmanagerConfig resource URI, including in the request body the declarative descriptor for the respective AlertmanagerConfig custom resource object. The payload should define the necessary alerting configuration parameters such as routes, receivers, inhibition rules, and templates to control how alerts are grouped, filtered, and sent.

2) The Notification Manager creates an individual AlertmanagerConfig as a Notification Management related NFV object.

3) The Notification manager returns a "201 Created" response to the API consumer and includes in the payload body a representation of the created AlertmanagerConfig as a Notification Management related NFV object.

4) The API consumer sends a POST request to the Notification Manager with the appropriate Alertmanager resource URI, including in the request body the declarative descriptor of the respective Alertmanager custom resource object.

5) The Notification Manager creates an individual Alertmanager as a Notification Management related NFV object that defines how alerts should be managed and routed by Alertmanager, based on the AlertmanagerConfig.

6) The Notification Manager returns a "201 Created" response to the API consumer and includes in the payload body a representation of the created Alertmanager as a Notification Management related NFV object.

Postcondition: Upon successful completion, the individual AlertmanagerConfig and Alertmanager as the requested Notification Manager-related NFV objects have been created.

Error handling: In case of failure, appropriate error information is provided in the response.

B.5 Sequence diagram for the Metrics Analyser profiled solution

B.5.1 Flow of creating VMRule as a Metrics Analyser related NFV object

This clause describes a sequence for metrics analysis by creating an individual VMRule resource as a Metrics Analysis Exposure related NFV object.

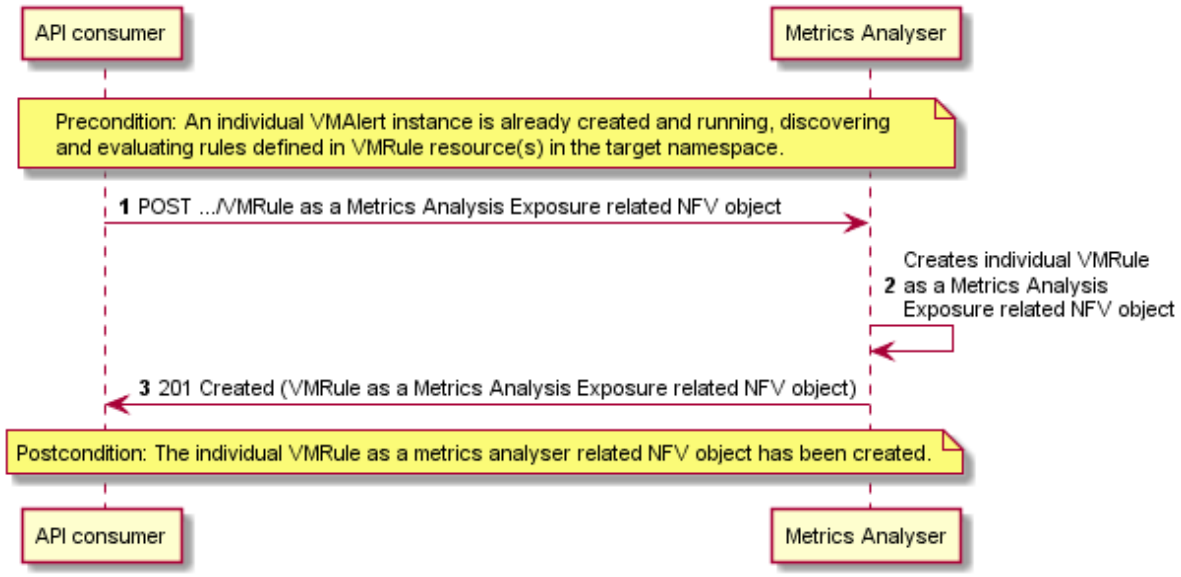

Figure B.5.1-1: Flow of metrics analysis by creating an individual VMRule resource as a Metrics Analysis Exposure related NFV object

The flow of analysing metrics consists of the following steps as illustrated in Figure B.5.1-1:

Precondition: An individual VMAlert instance is already created and running, discovering and evaluating rules defined in VMRule resource(s) in the target namespace.

1) The API consumer sends a POST request to the Metrics Analyser with the appropriate VMRule resource URI, including the data structure of the declarative descriptor of the respective VMRule custom resource object in the payload body.

2) The Metrics Analyser creates an individual VMRule as a Metrics Analysis Exposure related NFV object, specifying rules that define how metrics should be filtered, evaluated, and either recorded as new time-series data or used to trigger alerts based on defined expressions.

3) The Metrics Analyser returns a "201 Created" response to the API consumer and includes in the payload body a representation of the created VMRule as a Metrics Analysis Exposure related NFV object, which automatically enables VMAlert to begin evaluating the rules defined in the VMRule for real-time alerting or recording of metrics.

Postcondition: The individual VMRule as a metrics analyser related NFV object has been created.

Error handling: In case of failure, appropriate error information is provided in the response.

B.5.2 Flow of executing VictoriaMetrics Query API

This clause describes a sequence for executing VictoriaMetrics Query API.

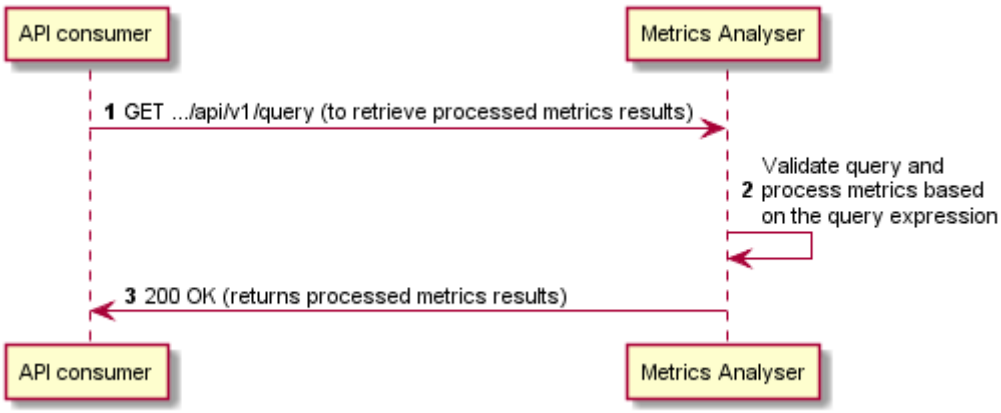

Figure B.5.2-1: Flow of executing VictoriaMetrics Query API

The flow of executing VictoriaMetrics Query API consists of the following steps as illustrated in Figure B.5.2-1:

Precondition: none.

1) The API consumer sends a GET (or POST) request to the Metrics Analyser using either "/api/v1/query" for retrieving metrics at a specific timestamp, or "/api/v1/query_range" for retrieving metrics over a defined time range. The request includes a MetricsQL expression as the query parameter, which can be provided as a URL query string for the GET method, or as URL-encoded form data in the request body for the POST method.

2) The Metrics Analyser validates the query expression, retrieves the relevant time-series data from storage, and processes it by applying the specified criteria defined in the MetricsQL expression.

3) The Metrics Analyser returns a 200 OK response with the processed metrics in structured JSON format, either a single result per series (for instant queries) or a sequence of timestamped values (for range queries).

Postcondition: None.

Error handling: In case of failure, appropriate error information is provided in the response.

Annex C (informative): Change history

Date	Version	Information about changes
June 2024	V0.0.1	Early draft version implementing approved contributions 172r1 and 186r1
October 2024	V0.0.2	Implemented approved contributions: NFVSOL(24)000237 - SOL024 include other PaaS Services NFVSOL(24)000253 - SOL024 Update scope description NFVSOL(24)000351r1 - SOL024ed521 Adding Analysis of Log Aggregator Functional requirements and FluentBit and Fluentd capabilities NFVSOL(24)000352r1 - SOL024ed521 Adding Analysis of Log Aggregator Functional requirements and Loki and OpenSearch capabilities NFVSOL(24)000353r1 - SOL024ed521 Adding Analysis of Log Aggregator Functional requirements and Open Telemetry Collector capabilities NFVSOL(24)000354r1 - SOL024ed521 Adding Analysis of Log Analyser Functional requirements and ElastAlert and Coroot capabilities NFVSOL(24)000355r1 - SOL024ed521 Adding Analysis of Log Analyser Functional requirements and Grafana and OpenSearch capabilities NFVSOL(24)000356r2 - SOL024ed521 Adding Analysis of Traffic Enforcer Functional requirements and Cilium and Istio capabilities NFVSOL(24)000357r2 - SOL024ed521 Adding Analysis of Traffic Enforcer Functional requirements and Linkerd and Envoy capabilities
November 2024	V0.0.3	Implemented approved contributions: NFVSOL(24)000393 - SOL024ed521 Update heading names in Annex_A NFVSOL(24)000415 - SOL024ed521 Comparison of the considered open-source solutions against Log Aggregator function interface requirements NFVSOL(24)000416 - SOL024ed521 Comparison of the considered open-source solutions against Log Analyser function interface requirements NFVSOL(24)000417 - SOL024ed521 Comparison of the considered open-source solutions against Traffic Enforcer function interface requirements NFVSOL(24)000422 - SOL024ed521 Update ElastAlert to ElastAlert 2 NFVSOL(24)000419r1 - SOL024ed521 Cross comparison of considered open-source solutions against Log Aggregator function NFVSOL(24)000420r1 - SOL024ed521 Cross comparison of considered open-source solutions against Log Analyser function NFVSOL(24)000421r1 - SOL024ed521 Cross comparison of considered open-source solutions against Traffic Enforcer function NFVSOL(24)000418r1 - SOL024ed521 Comparison of open-source solutions against VNF generic OAM functions functional and interface requirements Also few rapporteur actions (e.g. fix typos, correct font color, etc.)
December 2024	V0.0.4	Implemented approved contributions: NFVSOL(24)000441 -SOL024 Adding Analysis of Policy Agent Functional requirements and Open Policy Agent capabilities NFVSOL(24)000443r3 -SOL024 Adding Traffic Enforcer Object to map with profiled solution objects NFVSOL(24)000444r1 -SOL024 Adding Data Model section for the Traffic Management interface NFVSOL(24)000475 -SOL024 Adding Istio overview API structure and data model concepts NFVSOL(24)000476r1 - SOL024 Adding Traffic Management Interface based on the profiled solution NFVSOL(24)000478r1 - SOL024 Adding Resources for Traffic Management interface NFVSOL(24)000479 - SOL024 Adding Fluent Bit overview API structure and data model concepts NFVSOL(24)000480r1 - SOL024 Adding Log Exposure Interface based on the profiled solution NFVSOL(24)000482r1 - SOL024 Adding Resources for Log Exposure interface

Date	Version	Information about changes
February 2025	V0.0.5	Implemented approved contributions: NFVSOL(24)000361r1 - SOL024 Adding Analysis of VNF Metrics Aggregator Functional requirements and Prometheus capabilities NFVSOL(25)000002r1 - SOL024 Adding Log Aggregator Object to map with profiled solution objects NFVSOL(25)000003r1 - SOL024 Adding Data Model section for the Log Exposure interface NFVSOL(25)000011 - SOL024 Adding Analysis of VNF Metrics Analyser Functional requirements and Coroot capabilities NFVSOL(25)000025 - SOL024 Adding Log Analyser Object to map with profiled solution objects NFVSOL(25)000031 - SOL024 Adding Resource for Log Analysis Exposure interface NFVSOL(25)000032 - SOL024 Minor bug fixes of the wordings NFVSOL(25)000026r2 - SOL024 Adding Data Model section for the Log Analysis Exposure interface NFVSOL(25)000027r1 - SOL024 Adding OpenSearch overview API structure and data model concepts NFVSOL(25)000028r1 - SOL024 Adding Log Analysis Exposure Interface based on the profiled solution
March 2025	V0.0.6	Implemented approved contributions: NFVSOL(24)000465r1 - SOL024 Adding Analysis of VNF Metrics Aggregator Functional requirements and OTEL capabilities NFVSOL(24)000477r2 - SOL024 Adding Sequence diagrams for the Traffic Management interface NFVSOL(25)000048r1 - SOL024 Fix resource URIs and some other minor fixes NFVSOL(25)000047 - SOL024 Adding example CRD schemas for the OpenSearch Resource NFVSOL(25)000046 - SOL024 Adding example CRD schemas for the Fluent Bit Resources NFVSOL(25)000045 - SOL024 Adding example CRD schemas for the Istio Resources NFVSOL(25)000042r2 - SOL024_Adding_Annex_section_for_the_NotificationManager NFVSOL(25)000060r1 SOL024_Adding_Annex1&2_section_for_the_NotificationManager NFVSOL(25)000061r1 - SOL024_Adding_Annex3&4_section_for_the_NotificationManager NFVSOL(25)000073 - SOL024 Comparison of the considered open source solutions against VNF Metrics Aggregator interface requirements NFVSOL(25)000074 - SOL024 Adding analysis of VNF Metrics Analyser Functional requirements and OpenSearch capabilities NFVSOL(25)000075r1 - SOL024 Adding analysis of VNF Metrics Analyser Functional requirements and Prometheus Alertmanager capabilities NFVSOL(25)000076 - SOL024 Comparison of the considered open source solutions against VNF Metrics Analyser Interface requirements NFVSOL(25)000077r1 - SOL024 Comparison of the considered open source solutions against VNF Metrics Analyser Functional and Interface requirements NFVSOL(25)000078 - SOL024 Fix table numbering in the Annex NFVSOL(25)000085r1 - SOL024 Add note for FluentBit Output CRD resource Also few rapporteur actions (e.g. fix typos, etc.)

Date	Version	Information about changes
April 2025	V0.0.7	Implemented approved contributions: NFVSOL(24)000481r2 SOL024 Adding Sequence diagrams for the Log Exposure interface NFVSOL(25)000095 SOL024 Update Annex section for the NotificationManager InterfaceRequirement NFVSOL(25)000096r1 SOL024 Adding 4.2.3 section for the NotificationManager NFVSOL(25)00092r1 SOL024 Adding Sequence diagrams for the Log Analysis Exposure Interface NFVSOL(25)000107 SOL024 Adding Cilium overview API structure and data model concepts NFVSOL(25)000084r2 SOL024 Update Log Analysis Exposure interface to add PPL Query API NFVSOL(25)000117 SOL024 Adding 5.4 section for the NotificationManager NFVSOL(25)000116r2 SOL024 Adding 4.2.3 section for the NotificationManager Update NFVSOL(25)000124 SOL024 Minor bug fixes and Abbreviations update NFVSOL(25)000133 SOL024 Adding section for the NotificationManager sequence NFVSOL(25)000132r1 SOL024 Update Section19 NotificationManager CRDMapping As usual, few rapporteur actions (e.g. fix typos, etc.)
May 2025	V0.0.8	Implemented approved contributions: NFVSOL(25)000108_SOL024_Adding_Traffic_Enforcer_Object_to_map_with_Cilium_obj NFVSOL(25)000142_SOL024_Adding_5_4_section for the NotificationManager NFVSOL(25)000143_SOL024_One_section_of_Profiled_protocols_and_data_models NFVSOL(25)000144_SOL024_Updates_to_improve_readability_and_fix_several_issues NFVSOL(25)000145r3_SOL024_updatePrometheus_Alertmanager_to_Prometheus_to_profil NFVSOL(25)000158_SOL024_Cross-comparison_of_open-source_solutions_for_Metrics NFVSOL(25)000159_SOL024_Clause_4_amendments_and_releases_upgrades NFVSOL(25)000161r2_SOL024_Adding_Metrics_Exposure_Interface_based_on_the_profil NFVSOL(25)000162r1_SOL024_Adding_Resources_for_Metrics_Analysis_Exposure_Interf NFVSOL(25)000163r1_SOL024_Addding_Metrics_Analyser_Interface_based_on_the_profil
June 2025	V0.0.9	Implemented approved contributions: NFVSOL(24)000462_SOL024_Adding_Analysis_of_PaaS_Service_Configuration_Server NFVSOL(25)000160r2_SOL024_Fix_resource_trees_URIs_and_text_readability NFVSOL(25)000164r2_SOL024_Adding_Data_Model_section_for_the_Metrics_Analysis_Ex NFVSOL(25)000176_SOL024_Adding_sequence_diagrams_for_the_Metrics_Analysis_Exp NFVSOL(25)000177r1_SOL024_Add_VMRule_Example_CRD_Schema Rapporteur change to align the text in clause 2.2 as per the new ETSI GS skeleton

Date	Version	Information about changes
June 2025	V0.0.10	Implemented approved contributions: NFVSOL(25)000203r2_SOL024_Resource_structures_and_URIs_fixes_for_Traffic_Manage.zip NFVSOL(25)000204r1_SOL024_Resource_structures_and_URIs_fixes_for_Log_Exposure_I.zip NFVSOL(25)000205r2_SOL024_Resource_structures_and_URIs_fixes_for_Log_Analysis_E.zip NFVSOL(25)000206r1_SOL024_Resource_structures_and_URIs_fixes_for_Notification_M.zip NFVSOL(25)000207_SOL024_Add_end_to_end_explanatory_figures_for_the_considered.zip NFVSOL(25)000208_SOL024_Fix_DestinationRule_and_AuthorizationPolicy_CRD_schem.docx NFVSOL(25)000209_SOL024_Fix_OpenSearchCluster_CRD_schema.docx NFVSOL(25)000210_SOL024_Fix_AlertmanagerConfig_and_Alertmanager_CRD_schemas.docx NFVSOL(25)000211_SOL024_Fix_ClusterFilter_and_ClusterOutput_CRD_schemas.docx NFVSOL(25)000212_SOL024_update_Log_Aggregator_sequence_diagram.zip NFVSOL(25)000226r1_SOL024_Comparison_of_VNF_Metrics_Aggregator_functional_requi.docx NFVSOL(25)000227r1_SOL024_Resolve_Editors__Notes.docx NFVSOL(25)000236_SOL024_Update_note_in_section_10_1.docx NFVSOL(25)000237r1_SOL024_Update_section_4_overview_section.docx
July 2025	V0.0.11	Implemented approved contributions: NFVSOL(25)000257_SOL024_fix_minor_inconsistencies.docx
August 2025	V0.0.12	Fixing hanging paragraphs, use of shall in informative part of the deliverable, use of must, editorial modifications

History

Document history		
V5.3.1	September 2025	Publication