Axoflow

Axoflow documentation

Version 0.90.0

27 October 2026

Built from main 36974622

Contents

1 **Axoflow architecture** 11

2 **Try for free** 20

3 **Getting started** 21

3.1 Getting started using AxoStore 22

3.2 Getting started using a SIEM 40

4 **Deployment scenarios** 58

4.1 AxoConsole deployment 59

4.2 Transparent router mode 60

4.3 Router and edge deployment 62

4.4 AxoRouter high availability 64

4.5 Onboard existing syslog/syslog-ng infrastructure 66

4.6 Multi-cloud and multi-datacenter scenario 69

4.7 On-prem or air-gapped deployment 71

5 **Concepts** 74

5.1 Processing elements 75

5.2 Automatic data processing 76

5.3 Host attribution and inventory 80

5.4 Policy based routing 83

5.5 Classify and reduce security data 85

5.6 Normalization 89

5.7 Reliable transport 91

6 **Provision pipeline elements** 92

6.1 AxoRouter 93

6.1.1 Install AxoRouter on Linux 94

6.1.1.1 Advanced installation options 109

6.1.1.2 Run AxoRouter as non-root 131

6.1.1.3 Uninstall AxoRouter from Linux 133

6.1.2 Install AxoRouter on Kubernetes 137

6.1.2.1 Advanced installation options 145

6.1.3 Set up AxoRouter high availability with keepalived 150

6.2 AxoSyslog 158

6.3 Splunk Connect for Syslog (SC4S) 165

6.4 syslog-ng 167

6.5 Axolet 177

6.5.1 Install Axolet 178

6.5.2 Advanced installation options 185

6.5.3 Run axolet as non-root 191
6.5.4 Uninstall Axolet 193
6.5.5 Reinstall Axolet 195
6.6 Agent for Linux edge hosts 196
6.6.1 Manage the Linux agent 201
6.6.2 Advanced installation options 203
6.6.3 Configure data collection 208
6.6.4 Uninstall the Linux agent 209
6.7 Windows host - agent based solution 211
6.7.1 Manage the Windows agent 217
6.7.2 Advanced installation options 219
6.7.3 Configure data collection 225
6.7.4 Uninstall the Windows agent 226

7 Sources 228

7.1 Generic tips 229
7.2 Edge collection rules 235
7.2.1 File Collector 242
7.2.2 Journald Collector 248
7.2.3 Windows Event Log 254
7.2.4 Windows Event Tracing 261
7.3 Data forwarding from edge hosts 267
7.4 AxoRouter connector rules 272
7.5 OpenTelemetry 280
7.6 Syslog 286
7.7 Webhook 297
7.8 Generic HTTP 304
7.9 Elasticsearch Bulk API 312
7.10 Splunk HEC Server 320
7.11 Windows Event Collector (WEC) 328
7.12 Vendors 339
7.12.1 A10 Networks 342
7.12.1.1 vThunder 342
7.12.2 Amazon 344
7.12.2.1 AWS CloudTrail 344
7.12.2.2 AWS VPC Flow Logs 350
7.12.2.3 CloudWatch 356
7.12.3 Anthropic 363
7.12.3.1 Claude Code 363
7.12.4 Axoflow 365
7.12.4.1 AxoSyslog 365
7.12.5 Balasys 366

7.12.5.1 Proxedo Network Security 366
7.12.6 Broadcom 368
7.12.6.1 Edge Secure Web Gateway (Edge SWG) 368
7.12.6.2 Email Security 370
7.12.6.3 ESX 372
7.12.6.4 NSX 374
7.12.6.5 SSL Visibility Appliance 376
7.12.6.6 vCenter 378
7.12.7 Check Point 380
7.12.7.1 Anti-Bot 380
7.12.7.2 Anti-Malware 382
7.12.7.3 Anti-Phishing 384
7.12.7.4 Anti-Spam and Email Security 386
7.12.7.5 CPMI Client 388
7.12.7.6 cpmidu_update_tool 390
7.12.7.7 Database Tool 392
7.12.7.8 Endpoint Compliance 394
7.12.7.9 Endpoint Management 396
7.12.7.10 Forensics 398
7.12.7.11 GO Password Reset 400
7.12.7.12 HTTPS Inspection 402
7.12.7.13 iOS Profiles 404
7.12.7.14 IPS 406
7.12.7.15 MDS Query Tool 408
7.12.7.16 Media Encryption & Port Protection 410
7.12.7.17 Mobile Access 412
7.12.7.18 Next-Generation Firewall (NGFW) 414
7.12.7.19 QoS 416
7.12.7.20 Quantum 418
7.12.7.21 Query Database 419
7.12.7.22 SmartConsole 421
7.12.7.23 SmartUpdate 423
7.12.7.24 Syslog Data 425
7.12.7.25 Threat Emulation and Anti-Exploit 427
7.12.7.26 URL Filtering 429
7.12.7.27 Web API 431
7.12.8 Cisco 433
7.12.8.1 Access Control System (ACS) 433
7.12.8.2 Adaptive Security Appliance (ASA) 435
7.12.8.3 Application Control Engine (ACE) 437
7.12.8.4 Cisco IOS 439
7.12.8.5 Digital Network Architecture (DNA) 441
7.12.8.6 Email Security Appliance (ESA) 443

7.12.8.7 Firepower 446
7.12.8.8 Firepower Threat Defence (FTD) 448
7.12.8.9 Firewall Services Module (FWSM) 450
7.12.8.10 HyperFlex (HX, UCSH) 452
7.12.8.11 Identity Services Engine (ISE) 454
7.12.8.12 Integrated Management Controller (IMC) 456
7.12.8.13 IOS XR 458
7.12.8.14 Meeting Management 460
7.12.8.15 Meraki MX 462
7.12.8.16 Private Internet eXchange (PIX) 464
7.12.8.17 TelePresence Video Communication Server (VCS) 466
7.12.8.18 Unified Computing System Manager (UCSM) 468
7.12.8.19 Unified Communications Manager (UCM) 470
7.12.8.20 Viptela 472
7.12.9 Citrix 474
7.12.9.1 Netscaler 474
7.12.10 Cohesity 476
7.12.10.1 DataProtect 476
7.12.11 Corelight 478
7.12.11.1 Open Network Detection & Response (NDR) 478
7.12.12 CrowdStrike 481
7.12.12.1 CrowdStrike Falcon Platform 481
7.12.13 CyberArk 488
7.12.13.1 Privileged Threat Analytics (PTA) 488
7.12.13.2 Vault 490
7.12.14 Elastic 492
7.12.14.1 Beats, Elastic Agent, Logstash 492
7.12.14.2 Elasticsearch 493
7.12.15 Everpure 499
7.12.15.1 FlashArray 499
7.12.16 F5 Networks 501
7.12.16.1 BIG-IP 501
7.12.17 Forcepoint 504
7.12.17.1 Email Security 504
7.12.17.2 Next-Generation Firewall (NGFW) 506
7.12.17.3 WebProtect 508
7.12.18 Fortinet 510
7.12.18.1 FortiGate firewalls 510
7.12.18.2 FortiMail 515
7.12.18.3 FortiProxy 517
7.12.18.4 FortiWeb 519
7.12.19 Fortra 521
7.12.19.1 Powertech SIEM Agent for IBM i 521

7.12.20 General Unix/Linux host 523
7.12.20.1 Generic Linux services 523
7.12.21 Imperva 526
7.12.21.1 Incapsula 526
7.12.21.2 SecureSphere 528
7.12.22 Infoblox 530
7.12.22.1 NIOS 530
7.12.23 Ivanti 532
7.12.23.1 Connect secure 532
7.12.24 Juniper 534
7.12.24.1 Junos OS 534
7.12.25 Kaspersky 536
7.12.25.1 Endpoint Security 536
7.12.26 Kubernetes 538
7.12.26.1 NGINX Ingress 538
7.12.26.2 Telemetry Controller 539
7.12.27 Microsoft 540
7.12.27.1 Azure Event Hubs 540
7.12.27.2 Cloud App Security (MCAS) 547
7.12.27.3 Windows hosts 549
7.12.28 MikroTik 551
7.12.28.1 RouterOS 551
7.12.29 NetFlow Logic 553
7.12.29.1 NetFlow Optimizer 553
7.12.30 Netgate 559
7.12.30.1 pfSense 559
7.12.31 Netmotion 561
7.12.31.1 Netmotion 561
7.12.32 NETSCOUT 563
7.12.32.1 Arbor Edge Defense (AED) 563
7.12.32.2 Arbor Pravail (APS) 565
7.12.33 Omnissa 567
7.12.33.1 Horizon 568
7.12.34 OpenText 570
7.12.34.1 ArcSight 570
7.12.34.2 Self Service Password Reset (SSPR) 572
7.12.35 Palo Alto Networks 574
7.12.35.1 Cortex XSOAR 574
7.12.35.2 Idira 576
7.12.35.3 Palo Alto firewalls 581
7.12.36 Ping Identity 587
7.12.36.1 PingAccess 587
7.12.37 Progress 589

7.12.37.1 Flowmon Anomaly Detection System (ADS) 589
7.12.38 Riverbed 591
7.12.38.1 SteelConnect 591
7.12.38.2 SteelHead 593
7.12.39 RSA 595
7.12.39.1 Authentication Manager 595
7.12.40 rsyslog 597
7.12.41 SecureAuth 598
7.12.41.1 Identity Platform 598
7.12.42 Skyhigh Security 600
7.12.42.1 Secure Web Gateway 600
7.12.43 SonicWall 602
7.12.43.1 SonicWall 602
7.12.44 Splunk 611
7.12.44.1 Edge Processor 611
7.12.44.2 Heavy Forwarder 612
7.12.44.3 OpenTelemetry Collector 619
7.12.44.4 SC4S 620
7.12.45 Superna 621
7.12.45.1 Eyeglass 621
7.12.46 syslog-ng 626
7.12.47 Tanium 627
7.12.47.1 Tanium Platform 627
7.12.48 Tenable 629
7.12.48.1 Tenable One Vulnerability Management 629
7.12.49 Thales 634
7.12.49.1 Vormetric Data Security Platform 634
7.12.50 Trellix 636
7.12.50.1 Central Management System (CMS) 636
7.12.50.2 Email Threat Prevention (ETP) 638
7.12.50.3 Endpoint Security (HX) 640
7.12.50.4 ePolicy Orchestrator (EPO) 642
7.12.50.5 Intrusion Prevention System 644
7.12.50.6 MPS 646
7.12.51 Trend Micro 647
7.12.51.1 Deep Security Agent 647
7.12.52 Ubiquiti 649
7.12.52.1 Unifi 649
7.12.53 Varonis 651
7.12.53.1 DatAdvantage 651
7.12.54 Vectra AI 653
7.12.54.1 AI Platform 654
7.12.55 Zscaler appliances 656

7.12.55.1 Zscaler Nanolog Streaming Service 656
7.12.55.2 Zscaler Log Streaming Service 658

8 Destinations 660

8.1 Amazon 661
8.1.1 Amazon S3 661
8.2 Apache Kafka 669
8.3 AxoStore 670
8.4 ClickHouse 672
8.5 CrowdStrike 673
8.6 Dynatrace 683
8.7 Elastic 694
8.7.1 Elastic Cloud and Elasticsearch 694
8.8 Generic 706
8.8.1 /dev/null 706
8.8.2 syslog 710
8.9 Google 718
8.9.1 BigQuery 718
8.9.2 Google Cloud Pub/Sub 719
8.9.3 Google Security Operations (SecOps) 727
8.10 Grafana 738
8.10.1 Grafana Loki 738
8.11 Microsoft 739
8.11.1 Azure Monitor 739
8.11.2 Microsoft Sentinel 740
8.11.2.1 Configure Sentinel 751
8.12 OpenObserve 753
8.13 OpenSearch 765
8.14 Palo Alto Networks 766
8.14.1 Cortex XSIAM 766
8.15 Splunk 775
8.16 Sumo Logic 786

9 Storage 798

9.1 Overview 799
9.2 Search 802
9.3 Create Store 809
9.4 Create External Store 812
9.5 Create Logstore 815
9.6 Store metrics 822
9.7 Delete or Wipe Store 825
9.8 Rehydrate data 826

10 Data management 827
10.1 Flow overview 828
10.2 Manage flows 831
10.3 Processing steps 835
10.4 Flow metrics 854
10.5 Flow analytics 857
10.6 Flow tapping 858
10.7 Rehydrate data 868
10.8 AI assistant 872

11 Manage and monitor the pipeline 876
11.1 Dashboard 877
11.2 Topology 879
11.3 Hosts 884
11.3.1 Find a host 885
11.3.2 Host information 888
11.3.3 Custom labels and metadata 890
11.3.4 Bulk update hosts 892
11.3.5 Services 896
11.4 Log tapping 900
11.5 Alerts 908
11.5.1 Alerts reference 911
11.6 Private connections 927
11.6.1 Google Private Service Connect 927

12 Metrics and analytics 930
12.1 Analytics 931
12.2 Host metrics 940
12.3 Usage metrics 946

13 Integrations 947

14 Activity log 998

15 Troubleshooting Axoflow 1001
15.1 A service fills the disk with logs 1007

16 Access management 1014

17 Deploy AxoConsole 1018
17.1 On-premise 1019
17.1.1 Deployment guide 1019
17.1.2 Authentication 1027
17.1.2.1 Dex 1028

17.1.2.1.1 GitHub 1029
17.1.2.1.2 Google 1035
17.1.2.1.3 LDAP 1042
17.1.3 Authorization 1047
17.1.4 Prepare AxoRouter hosts 1050
17.1.5 Upgrade 1052
17.2 Air-gapped 1053

18 Axoflow reference 1064

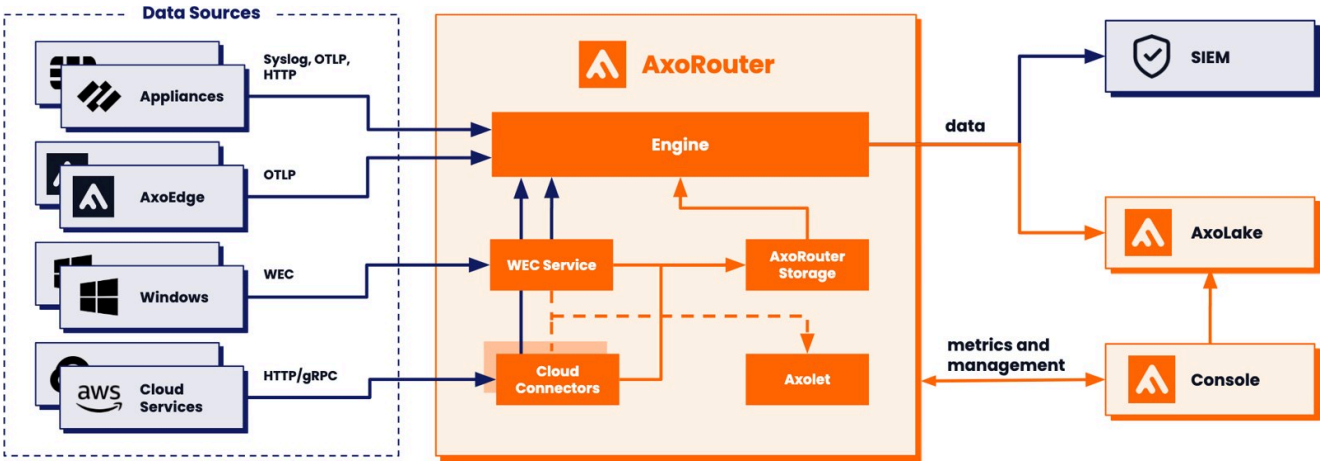
18.1 Global search 1065
18.2 AQL operator reference 1068
18.3 Glossary 1070
18.4 Manual pages 1071
18.4.1 axorouter-ctl 1071
18.5 Message schema 1076
18.5.1 Overview 1076
18.5.2 Message schema reference 1077
18.6 Metrics schema 1106
18.7 Storage schema 1121

19 Legal notice 1126

1 Axoflow architecture

The components of the Axoflow Platform, and how the control plane (AxoConsole) and the data plane (AxoRouter, AxoEdge, AxoStore) work together.

Axoflow Platform provides an end-to-end pipeline that automates the collection, management, and loading of your security data in a vendor-agnostic way. The following figure highlights the Axoflow data flow:



Control plane and data plane

Axoflow is split into two halves: a control plane that you configure and monitor the pipeline from, and a data plane that carries your security data. Only the data plane ever touches the data itself.

Control plane

The management half of Axoflow: it configures the pipeline and describes what flows through it, but never carries your data.
In practice, the control plane is AxoConsole. It collects the metrics that the pipeline components report and draws them as a live topology, so what it holds is the *metadata* of each event: the source it originated from, the size in bytes (and event count over time), its destination, and any other element that *describes* the data.

You also manage the pipeline from here. AxoConsole deploys the managed components, translates every flow you configure into component-level configuration, and raises alerts on anomalies, so you never have to edit agent configuration files by hand.

Data plane

The collector agents, processing engines, and storage that carry and process your telemetry data, as opposed to the control plane that manages them.
The data plane covers AxoRouter, the Axoflow agents, and the Axoflow Cloud Connectors. These components collect, classify, filter, and transform your telemetry data, deliver it to its proper destinations (SIEMs, storage), and report metrics to the control plane as they go. The storage your data lands in belongs on this side of the line too: AxoStore keeps data on the AxoRouter host, and AxoLake keeps it long term.

You can manage data plane components from the AxoConsole, or run them independently of it.

> **Note**
>
> **Note that AxoRouter and Axoflow agent collects detailed, real-time metrics about the data-flows – giving you observability over the health of the security data pipeline and its components. Your security data remains in your self-managed cloud or in your on-prem instance where your sources, destinations, Axoflow agents, and AxoRouters are running, only metrics are forwarded to AxoConsole.**

Pipeline components

A telemetry pipeline consists of the following high-level components:

- *Data Sources*: Data sources are the endpoints of the pipeline that generate the logs and other telemetry data you want to collect. For example, firewalls and other appliances, Kubernetes clusters, application servers can all be data sources. Data sources send their data either directly to a *destination*, or to a *router*.

  Axoflow provides several log collecting agents and solutions to collect data in different environments, including connectors for cloud services, Kubernetes clusters, Linux servers, and Windows servers.

- *Routers*: Router (also called relays or aggregators) collect the data from a set of data sources and transport them to the destinations.

  AxoRouter can collect, curate, and enrich the data: it automatically identifies your log sources and fixes common errors in the incoming data. It also converts the data into a format that best suits the destination to optimize ingestion speed and data quality.

- *Destinations*: Destinations are your SIEM and storage solutions where the telemetry pipeline delivers your security data.

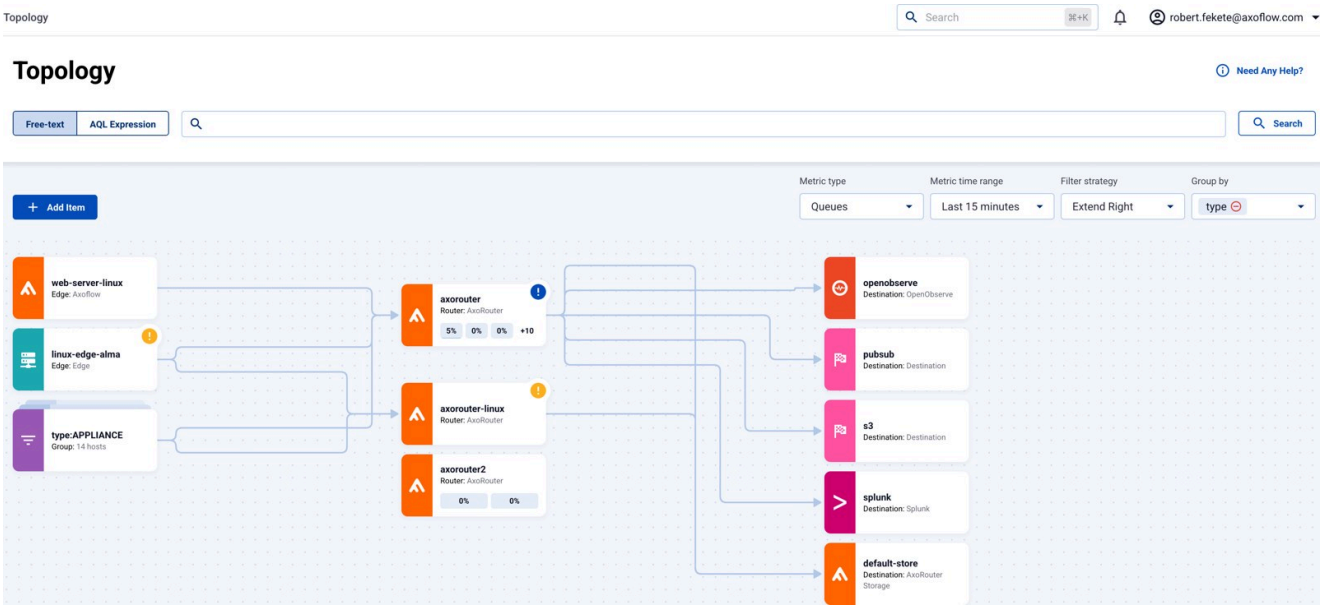


Your telemetry pipeline can consist of managed and unmanaged components. Axoflow provides several managed Axoflow agent components that help you collect or fetch data from your various data sources, or act as routers.

AxoConsole

AxoConsole (formerly Axoflow Console) is the control plane of Axoflow: the data visualization and management UI. Available both as a SaaS and an on-premises solution, including air-gapped and hybrid environments. AxoConsole collects and visualizes the metrics received from the pipeline components to provide insight into the details of your telemetry pipeline and the data it processes. It also allows you to:

- deploy managed components (like Axoflow agent and AxoRouter),
- configure data routing and processing flows that the AxoConsole automatically translates into specific configuration and deploys to the managed pipeline components, and
- troubleshoot your pipeline by tapping real-time into the various data flows and agent logs.

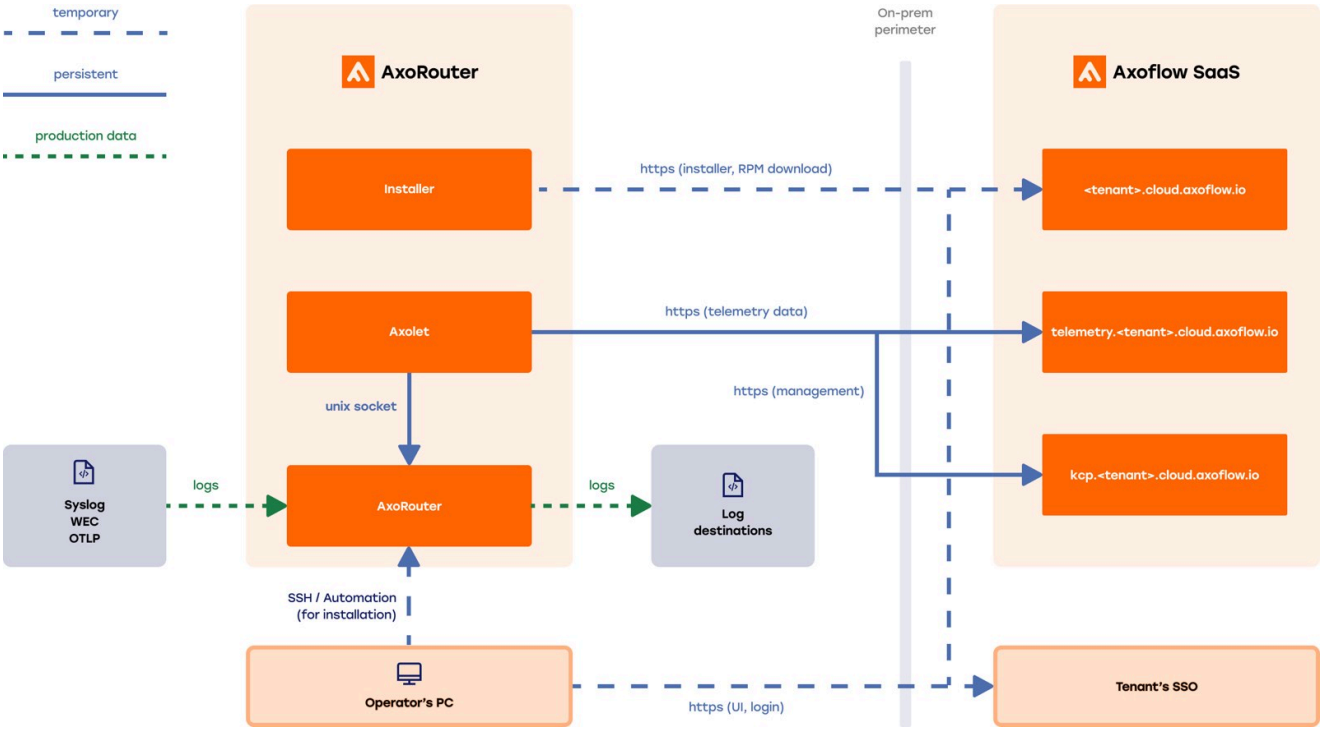


AxoRouter

AxoRouter is the main processing engine of the data plane: a router (aggregator) and data curation engine that collects all kinds of telemetry and security data and has all the low-level functions you would expect of log-forwarding agents and routers. AxoRouter can also curate and enrich the collected data, it:

- automatically identifies your log sources, recognizing the product that's sending it, for example, switches, firewalls, and web gateways
- fixes common errors in the incoming data: for example, it corrects missing hostnames, invalid timestamps, formatting errors, and so on
- automatically parses the logs of hundreds of COTS devices
- performs data reduction and normalization to reduce noise and improve data quality.

Before sending your data to its destination, AxoRouter automatically converts the data into a format that best suits the destination to optimize ingestion speed and data quality. For example, when sending data to Splunk, setting the proper sourcetype and index is essential.

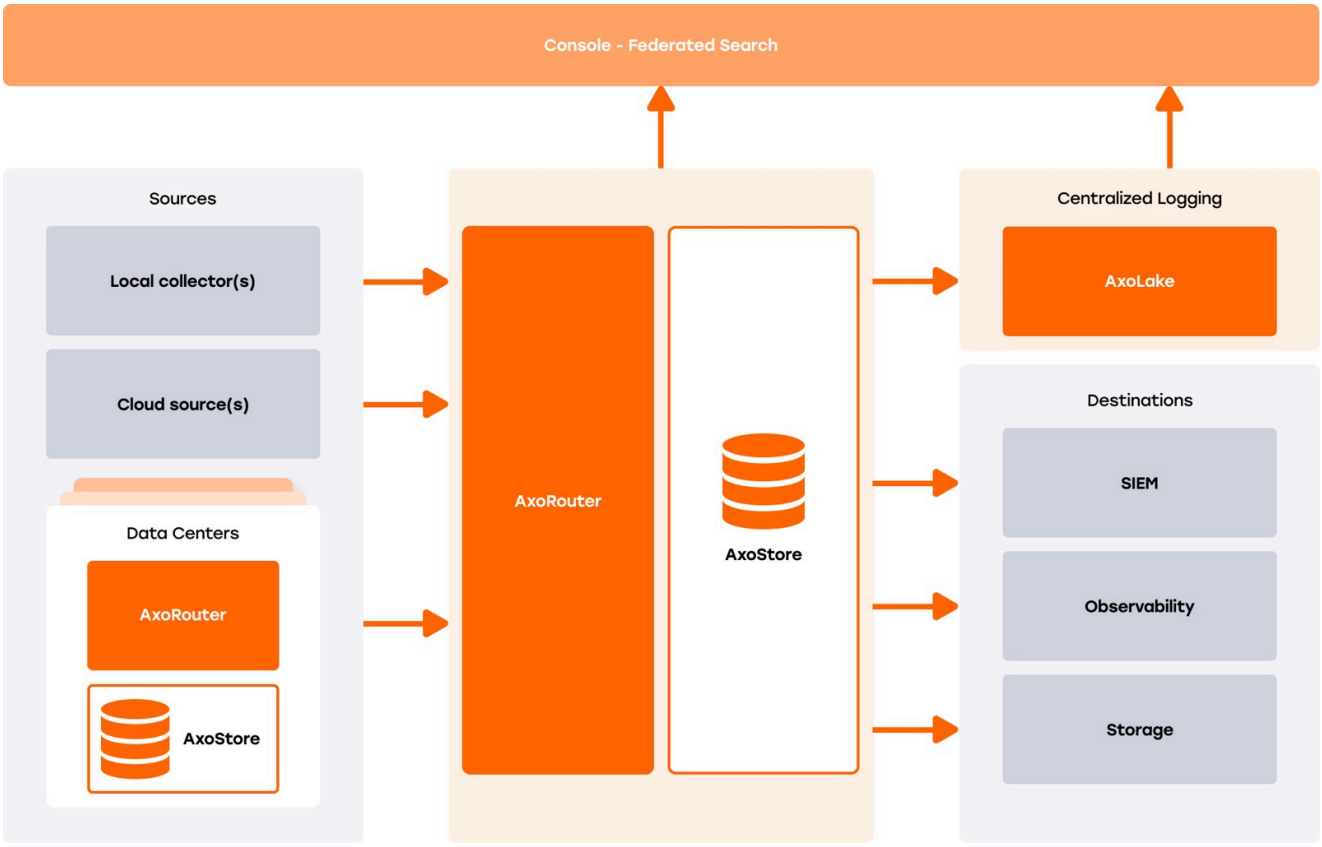


Axolet

Axolet connects the data plane to the control plane. It's a monitoring and management agent that integrates with the local log collector (like AxoSyslog, Splunk Connect for Syslog, or syslog-ng) that runs on the data source and provides detailed metrics about the host and its data traffic to the AxoConsole. AxoRouter deployments automatically include Axolet.

Storage solutions

Storage sits in the data plane, alongside the components that route the data into it. In addition to being able to route your data to various storage and SIEM solutions, Axoflow provides a range of storage solutions for your use cases, from small, temporal storage to petabyte-scale long-term data retention.



AxoStore

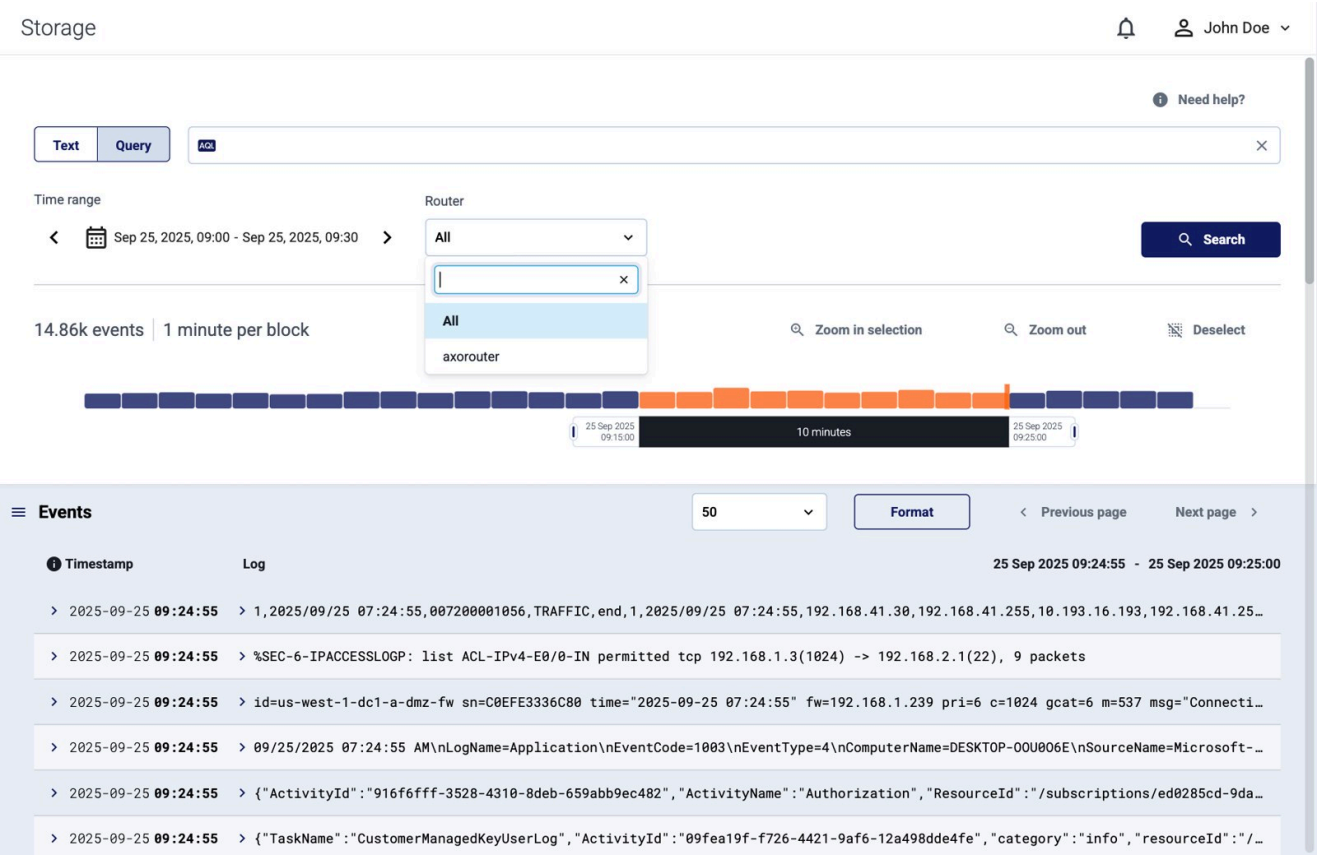
AxoStore is a lightweight, queryable, temporary storage option that stores data locally on the AxoRouter host. AxoConsole provides federated search for every AxoRouter that has AxoStore deployed, so you can simultaneously run queries on every AxoStore.

AxoStore is automatically available for all Axoflow customers, for each AxoRouter node, with 7 days retention, and up to 1TB of storage. For longer retention time or bigger storage, you need an extended AxoStore subscription. Contact us for the details.

You can retrieve selected data from AxoStore and (rehydrate) it to other destinations for further analysis as needed.

For example, AxoStore allows you to:

- Store debug-level logs for a short time (for example, 24h) to support “what if something happens” scenarios.
- Collect and search data ad-hoc for maintenance or debugging tasks.
- Keep a lightweight local backup of your security data.
- Retrieve selected data from AxoStore and (rehydrate) it to other destinations for further analysis as needed.



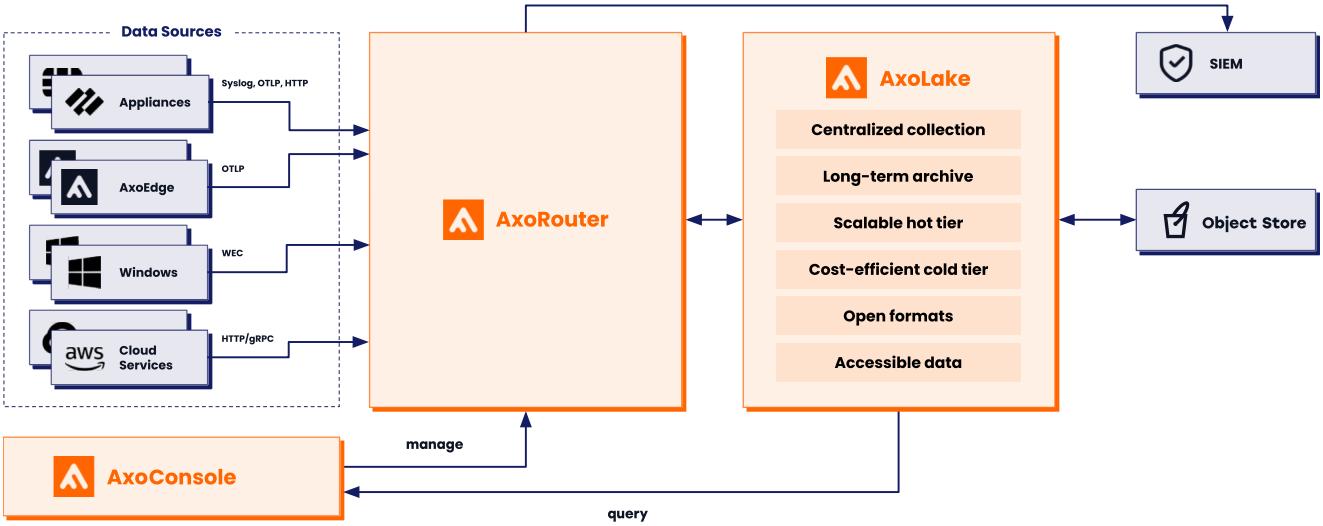
For details, see Storage.

AxoLake

AxoLake is a tiered SaaS security lake with a scalable hot tier for fast queries, and cost-efficient, S3-compatible cloud-based cold tier. You can use high-level, policy-based routing to decide which data goes to cold storage, hot storage, or your SIEM. If you need to work with data from cold storage, you can rehydrate it into hot storage, or even replay it to send it into your SIEM. Designed for long-term retention, open access, and integration with SIEMs and security analytics.

AxoLake provides cheap, long-term archive storage, while making hot data available for detection and analytics.

For details and pricing, see the AxoLake product page, or contact us.



Axoflow agents and collectors

Axoflow provides the following data plane agents and collectors:

- AxoRouter is the main collector and data processing engine for push-based protocols, such as Syslog, OTLP, HTTP, and WEC.
- Axoflow agent (AxoEdge) is a collection of managed agents for Linux, Kubernetes, and Microsoft Windows. They collect local data from the host they're deployed on, send it to an AxoRouter instance, and provide metadata for AxoConsole. Axoflow agent provides file and journald collectors on Linux and Event log and Event trace collectors on Windows.
- Axoflow Cloud Connectors are pull-based collectors built on top of OpenTelemetry that integrate with all modern Cloud Services (for example, AWS CloudWatch, Amazon S3, GCP Pub/Sub, Azure Event Hub), API and streaming sources (CrowdStrike, Kafka) and legacy methods (SQL) for enterprise log collection use cases. In addition to the existing sources, Axoflow Cloud Connectors can consume events from a vast amount of HTTP or streaming APIs and can be extended by the Axoflow team to support any custom pull based HTTP or streaming API source. A few specific examples:

Receiver	Cloud source
`awscloudwatchreceiver`	AWS CloudWatch Logs
`awsfirehosereceiver`	Kinesis Firehose
`awss3receiver`	S3 log storage
`awslambdareceiver`	Lambda telemetry
`azureblobreceiver`	Azure Blob Storage

`azureeventhubreceiver`	Azure Event Hub
`cloudflarereceiver`	Cloudflare Logpush
`cloudfoundryreceiver`	Cloud Foundry logs
`googlecloudpubsubreceiver`	GCP Pub/Sub
`kafkareceiver`	Kafka log streams
`lokireceiver`	Loki / Promtail
`fluentforwardreceiver`	Fluentd / Fluent Bit
`syslogreceiver`	Syslog (cloud services)

2 Try for free

Try Axoflow for free: watch product videos, request a demo sandbox, a free evaluation version, or a live demo.

To see Axoflow in action, you have a number of options:

- Watch some of the product videos on our blog.
- Request a demo sandbox environment. This pre-deployed environment contains a demo pipeline complete with destinations, routers, and sources that generate traffic, and you can check the metrics, use the analytics, modify the flows, and so on to get a feel of using a real Axoflow deployment.
- Request a free evaluation version. That's an empty cloud deployment you can use for testing and PoC: you can onboard your own pipeline elements, add sources and destinations, and see how Axoflow performs with your data!
- Request a live demo where our engineers show you how Axoflow works, and can discuss your specific use cases in detail.

3 Getting started

First steps with Axoflow: deploy an AxoRouter and send data to a local AxoStore or to Splunk or a similar SIEM.

Select in which scenario you want to start using Axoflow:

- Deploy an AxoRouter and send data into a local AxoStore: see Getting started using AxoStore.
- Deploy an AxoRouter and send data to Splunk or a similar SIEM: see Getting started using a SIEM.

To jump to any page, object, or integration in AxoConsole, select `⌘/Ctrl + K`. For details, see Global search.

To see all the sources, connector rules, collection rules, and destinations that Axoflow supports, and to add them, see Integrations.

3.1 Getting started using AxoStore

Install AxoRouter, connect a source, store its data in AxoStore through your first flow, and search the stored logs.

This guide shows you how to get started with Axoflow, and use the built-in AxoStore for destination. You're going to install AxoRouter, and configure or create a source to send data to AxoRouter. You'll also configure AxoRouter to forward the received data to AxoStore, a temporary storage solution available on the AxoRouter host. The resulting topology will look something like this:



Why use Axoflow

Using the Axoflow security data pipeline automatically corrects and augments the security data you collect, resulting in high-quality, curated, SIEM-optimized data. It also removes redundant data to reduce storage and SIEM costs. In addition, it allows automates pipeline configuration and provides metrics and alerts for your telemetry data flows.

Prerequisites

You'll need:

- An Axoflow subscription, access to a free evaluation version, or an on-premise deployment.
- A data source. This can be any host that you can configure to send syslog or OpenTelemetry data to your AxoRouter instance that you'll install.
- A host that you'll install AxoRouter on. This can be a separate Linux host, or a virtual machine running on your local computer.

  AxoRouter supports the following operating systems:

  - Ubuntu 24.04 and 26.04,
  - Red Hat Enterprise Linux 8.8 or newer, 9.4 or newer, and 10.

  Other compatible and derivative distributions (for example, AlmaLinux or Rocky Linux) should work as well, but we don't test them explicitly.

Network access

The host must be able to access the following domains related to the AxoConsole:

- When using AxoConsole SaaS:
  - `<your-tenant-id>.cloud.axoflow.io` : HTTPS traffic on TCP port 443, needed to download the binaries for Axoflow software (like Axolet and AxoRouter).
  - `kcp.<your-tenant-id>.cloud.axoflow.io` : HTTPS (mutual TLS) traffic on TCP port 443 for management traffic.
  - `telemetry.<your-tenant-id>.cloud.axoflow.io` : HTTPS (mutual TLS) traffic on TCP port 443, where Axolet sends the metrics of the host.
  - `us-docker.pkg.dev` : HTTPS traffic on TCP port 443, but only if you're forcing the installation script to pull the container images from the public repository. The Axolet and AxoRouter for Linux installation scripts download the images directly from AxoConsole.
- When using an on-premise AxoConsole:
  - The following domains should point to AxoConsole IP address to access Axoflow from your desktop and AxoRouter hosts:
    - `your-host.your-domain` : The main domain of your AxoConsole deployment.
    - `idp.your-host.your-domain` : A subdomain for the identity provider.
    - `kcp.your-host.your-domain` : A subdomain for management traffic.
    - `telemetry.your-host.your-domain` : A subdomain for agent telemetry.
  - The AxoConsole host must have the following **Open Ports**:
    - 80 TCP: Traefik ingress (HTTP, redirected to HTTPS).
    - 443 TCP: HTTPS for the AxoConsole UI, API, and all web traffic.
- When installing Axoflow agent for Windows or Linux:
  - `github.com` : HTTPS traffic on TCP port 443, for downloading installer packages.

Log in to the AxoConsole

Verify that you have access to the AxoConsole.

1. Open `https://<your-tenant-id>.axoflow.io/` in your browser.
2. Log in using Google Authentication.

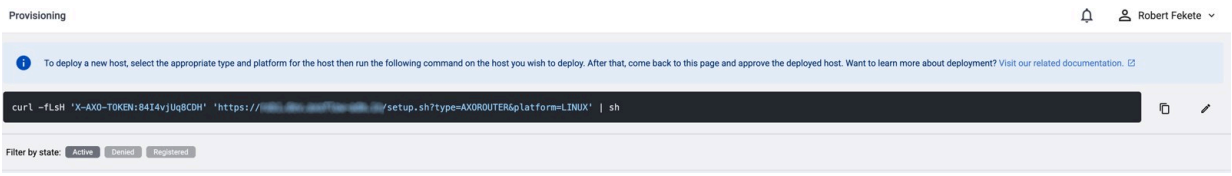
Deploy an AxoRouter instance

Deploy an AxoRouter instance that will route, curate, and enrich your log data.

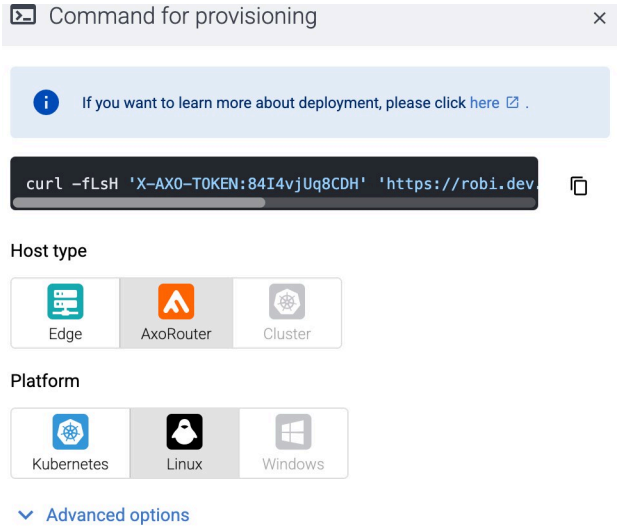
> ## Note
>
> **Note that AxoRouter and Axoflow agent collects detailed, real-time metrics about the data-flows – giving you observability over the health of the security data pipeline and its components. Your security data remains in your self-managed cloud or in your on-prem instance where your sources, destinations, Axoflow agents, and AxoRouters are running, only metrics are forwarded to AxoConsole.**

Deploy AxoRouter on Linux. For other platforms, see AxoRouter.

1. Select **Routers > Add Router**.



2. Select the platform (**Linux**). The one-liner installation command is displayed.



3. (Optional) If you don't want to store any logs locally on AxoRouter, disable AxoStore, select **Advanced options**, scroll down, and deselect **Enable AxoStore**.

4. (Optional)

   If needed, set the **Advanced options** (for example, proxy settings, or **External image pull** in air-gapped environments) to modify the installation parameters. Usually, you don't have to use advanced options unless the Axoflow support team instructs you to do so.

5. Open a terminal on the host where you want to install AxoRouter.

6. Run the one-liner, then follow the on-screen instructions.

> **Note**
>
> **Running the provisioning command with `sudo` would mask environment variables of the calling shell. Either start the whole procedure from a root shell, or let the install script call sudo when it needs to. In other words: don't add the `sudo` command to the provisioning command.**

Example output:

```
Do you want to install AxoRouter now? [Y]

% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  5480  100  5480    0     0  32076      0 --:--:-- --:--:-- --:--:--
33414
Selecting previously unselected package axorouter.
(Reading database ... 17697 files and directories currently installed.)
Preparing to unpack axorouter.deb ...
Unpacking axorouter (0.66.0) ...
Setting up axorouter (0.66.0) ...
Low maximum socket receive buffer size value detected: 7500000 bytes (7.2MB).
Do you you want to permanently set the net.core.rmem_max sysctl value to
33554432 bytes (32MB) on this system? [Y]

net.core.rmem_max = 33554432
Created symlink '/etc/systemd/system/multi-user.target.wants/axostore.path'
→ '/etc/systemd/system/axostore.path'.
Created symlink '/etc/systemd/system/multi-user.target.wants/axorouter-
wec.path' → '/etc/systemd/system/axorouter-wec.path'.
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 42.9M  100 42.9M    0     0  28.1M      0  0:00:01  0:00:01 --:--:--
28.2M
Selecting previously unselected package axolet.
(Reading database ... 17707 files and directories currently installed.)
Preparing to unpack axolet.deb ...
Unpacking axolet (0.66.0) ...
Setting up axolet (0.66.0) ...
Created symlink '/etc/systemd/system/multi-user.target.wants/axolet.service'
→ '/usr/lib/systemd/system/axolet.service'.
Now continue with onboarding the host on the Axoflow web UI.
```

Note

If you get the following errors:

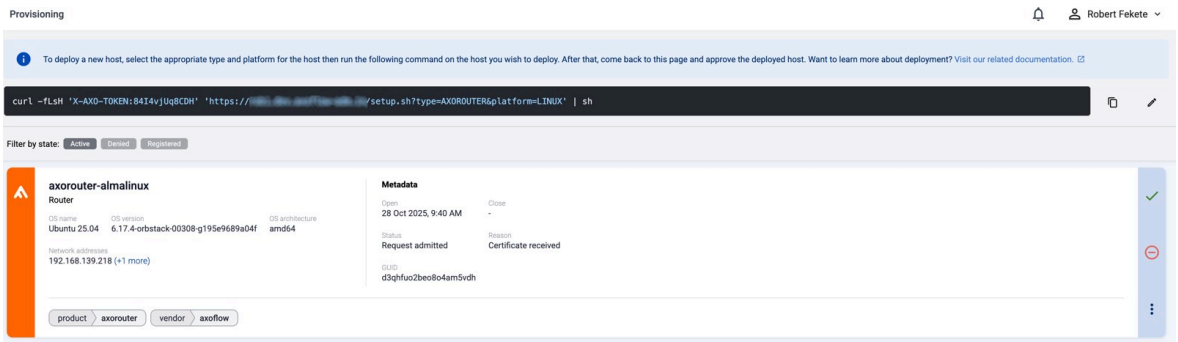
```
AXOROUTER_FUSE envvar is not set. It may mean that the
/usr/local/bin/setup-axorouter script is called without proper
configuration.
Failed to configure AxoRouter.
Failed to deploy the AxoRouter.
```

Set the following (or equivalent) in the `sudoers` file on the host, and rerun the installation command.

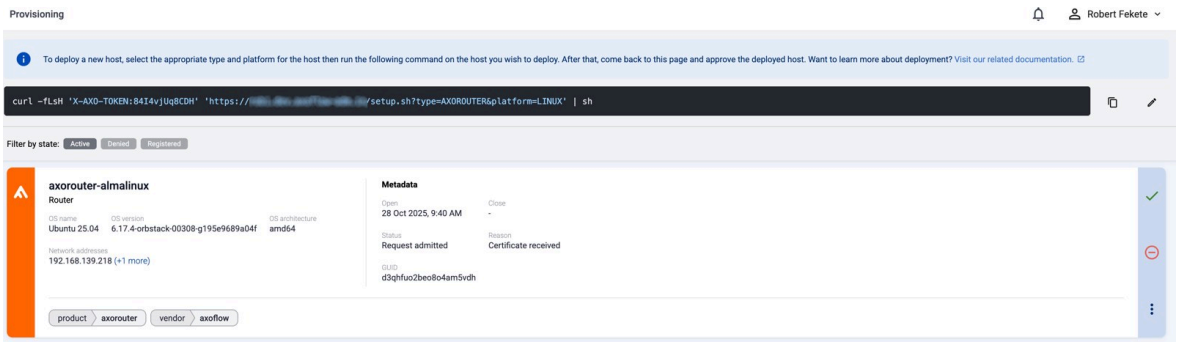
```
Defaults!/usr/local/bin/setup-axorouter env_keep += "AXO* http_proxy
https_proxy no_proxy"
Defaults!/usr/local/bin/axorouter-ctl env_keep += "AXO* http_proxy
https_proxy no_proxy"
Defaults!/usr/local/bin/configure-axolet env_keep += "AXO* http_proxy
https_proxy no_proxy"
```

7. Register the host.

    1. Reload the **Provisioning** page. There should be a registration request for the new AxoRouter deployment. Select ✓.



    2. Select **Register** to register the host. You can add a description and labels (in `label:value` format) to the host.



3. If the primary IP address (the first IP address shown in the **Network addresses** section on the **Routers** page for each AxoRouter) is not accessible from your edge hosts, set a **Network address override** (IP address or an FQDN) that's accessible. Otherwise, data forwarding from edge hosts will fail.

4. Select the **Topology** page. The new AxoRouter instance is displayed.

Add a source

Configure a host to send data to AxoRouter.

Configure a generic syslog host. For sources that are specifically supported by Axoflow, see Sources.

1. Log in to your device. You need administrator privileges to perform the configuration.

2. If needed, enable syslog forwarding on the device.

3. Set AxoRouter as the syslog server. Typically, you can configure the following parameters:

   - **Name or IP Address of the syslog server**: Set the address of your AxoRouter.
   - **Protocol**: If possible, set TCP or TLS.

     > **Note**
     >
     > **If you're sending data over TLS, make sure to configure a TLS-enabled connector rule in Axoflow.**

   - **Syslog Format**: If possible, set RFC5424 (or equivalent), otherwise leave the default.
   - **Port**: Set a port appropriate for the protocol and syslog format you have configured.

By default, AxoRouter accepts data on the following ports (unless you've modified the default connector rules):

- 514 UDP and TCP for RFC3164 (BSD-syslog) and RFC5424 (IETF-syslog) formatted traffic. AxoRouter automatically recognizes and handles both formats.
- 601 TCP (RFC 3195 - Reliable Delivery for syslog) for RFC5424 (IETF-syslog) and RFC3164 (BSD-syslog) formatted traffic. AxoRouter automatically recognizes and handles both formats.
- 6514 TCP for TLS-encrypted syslog traffic (RFC 5425).
- 4317 TCP for OpenTelemetry log data.
- 5085 TCP for the Windows Event Collector (WEC) service.
- 8088 TCP for the Splunk HEC receiver (optional, only if you forward Splunk HEC traffic to AxoRouter).
- 9200 TCP for the Elasticsearch Bulk API connector (optional, only if you forward data from Elastic agents to AxoRouter).

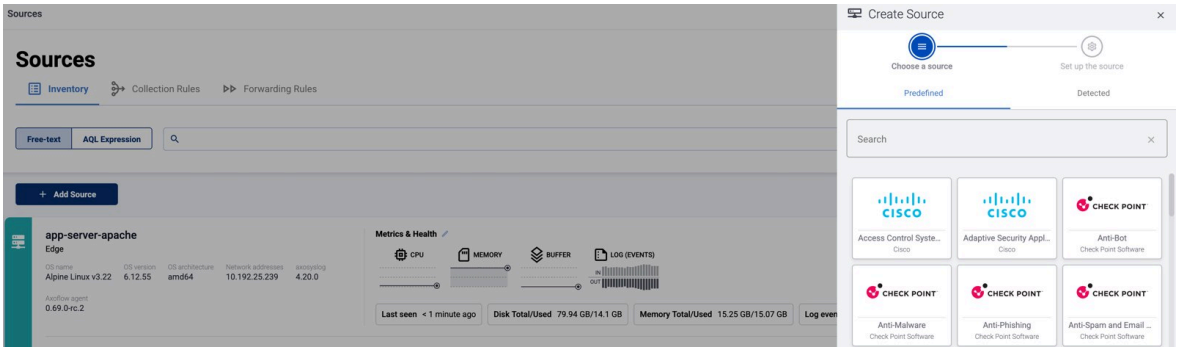
To receive data on other ports or other protocols, configure other connector rules for the AxoRouter host.

For TLS-encrypted syslog connections, create a new connector rule or edit an existing one, and configure the keys and certificates needed to encrypt the connections. For details, see Syslog.

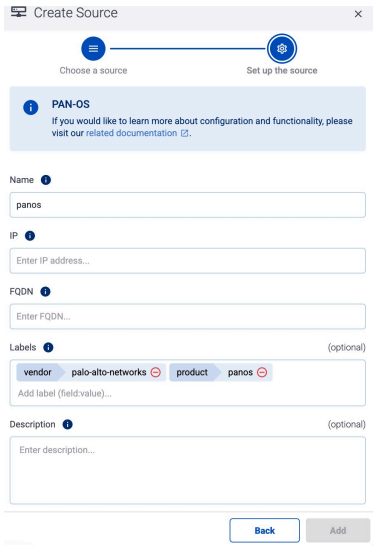
> **Note**
>
> **When you create a connector rule on a new port, open that port in the firewall of of every corresponding AxoRouter as well.**

4. Add the source to AxoConsole.

    1. Open the AxoConsole and select **Topology**.
    2. Select **Add Item > Source**.



- If the source is actively sending data to an AxoRouter instance, select **Detected**, then select your source(s) and select **Add Selected Source(s)**.
- Otherwise, select the vendor and product corresponding to your source from the **Predefined** sources, then enter the parameters of the source, like **IP address** and **FQDN**.



> ## Note
>
> **During log tapping, you can add hosts that are actively sending data to an AxoRouter instance by clicking Register source.**

3. (Optional) Add custom labels as needed.

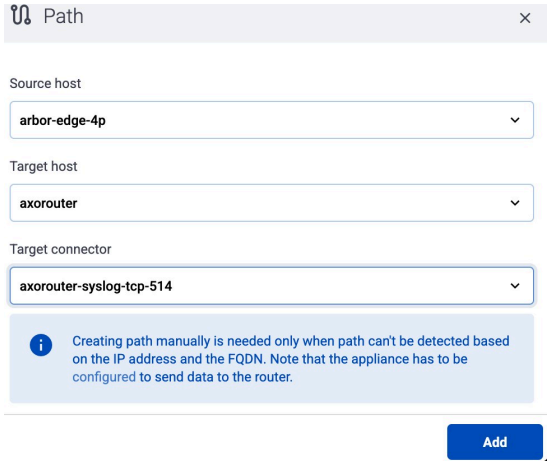
> **CAUTION:**
>
> AxoConsole can automatically raise an alert if a source that used to send data regularly stops sending data. To enable this alert for a source, add the `traffic_alerts:true` label to the source.

4. Select **Add**.

5. (Optional) **Add Path** manually. That's needed only when AxoConsole can't detect the path based on the IP address and the FQDN, and you haven't yet configured the source to send data to the router. If you've added the source from the **Detected** list, you can skip this step, as the path is created automatically.

> **Create a path between the source and the AxoRouter instance.**
>
> If you've added the source from the **Detected** list, you can skip this step, as the path is created automatically.

1. Select **Topology > Add Item > Path**.



2. If not set automatically, select the beginning of the path in the **Source host** field.
3. Select the target router or aggregator this source is sending its data to in the **Target host** field, for example, `axorouter`.
4. Select the **Target connector**. The connector determines how the destination receives the data (for example, using which protocol or port).
5. Select **Add**. The new path appears on the **Topology** page.



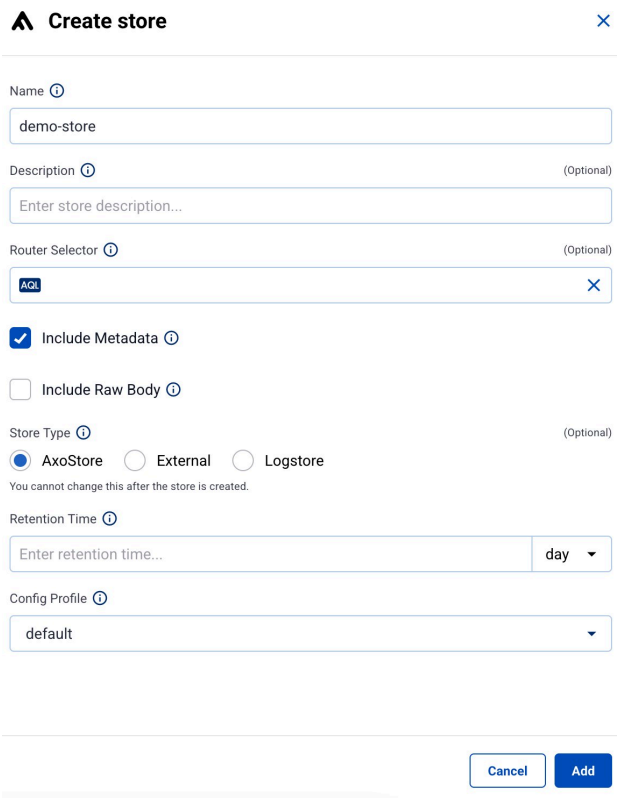
Note

If your syslog source is running `syslog-ng`, Splunk Connect for Syslog (SC4S), or AxoSyslog as its log forwarder agent, consider installing Axolet on the host and instrumenting the configuration of the log forwarder to receive detailed metrics about the host and the processed data. For details, see Manage and monitor the pipeline.

Create a store

Create a store called `default` to store the incoming messages.

1. Navigate to **Routers > Stores > Add Store**.
2. Enter `default` as the **Name** of the store. This name will appear in the list of destinations (with the `-store` suffix) when setting the destination of a Flow. You cannot modify the name of the store later.



3. Set **Store Type** to **AxoStore**.
4. Set the **Router Selector** so it matches the AxoRouter instances where you want to create this store.

   You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

   - If you leave the router selector empty, it matches every AxoRouter instance.
   - To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.
   - If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)
5. Set the **Retention Time** (in days) so older data is automatically deleted from the store. You cannot modify this parameter later.
6. Select **Add**.

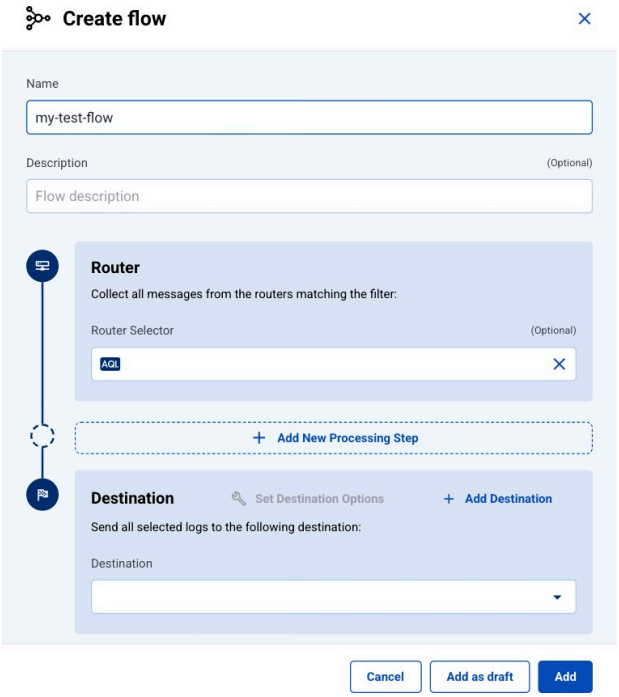
Create a flow

Create a flow to route the traffic from your AxoRouter instance to the `default-store` destination.

1. Select **Flows**.

2. Select **Add Flow > Flow**.

3. Enter a name for the flow, for example, `my-test-flow`.

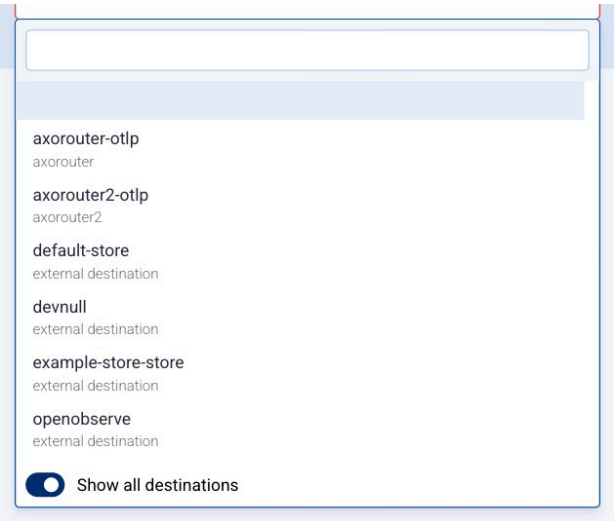


4. In the **Router Selector** field, enter an expression that matches the routers you want to apply the flow to. To select a specific router, use a name selector, for example, `name = my-axorouter-hostname`.

   You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

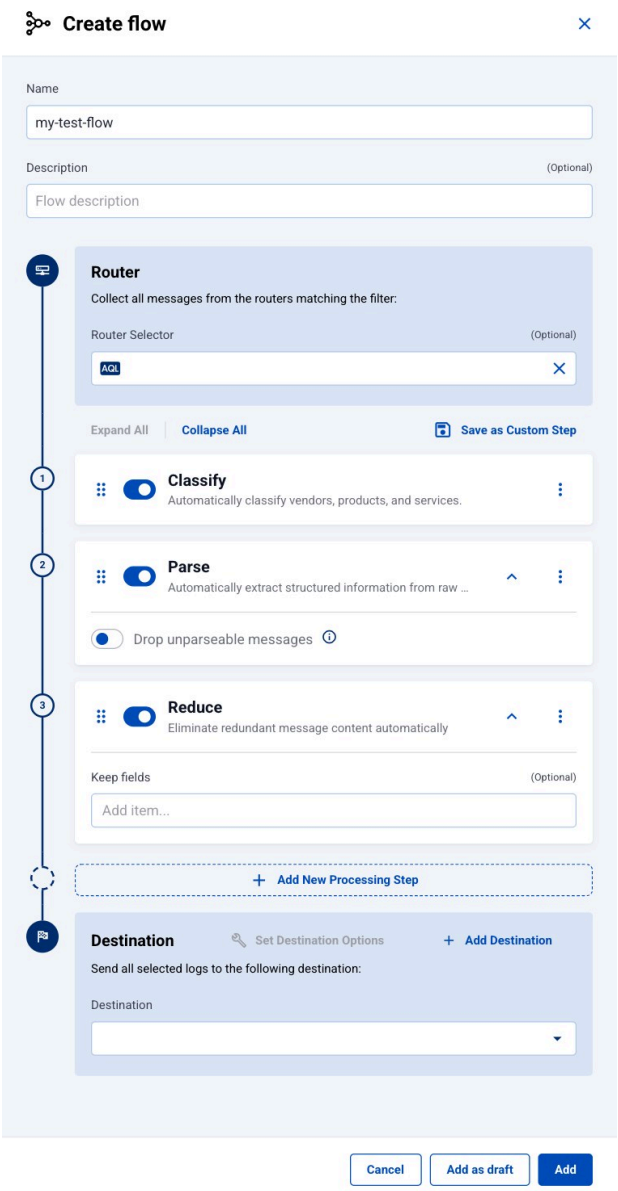
   - If you leave the router selector empty, it matches every AxoRouter instance.
   - To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.
   - If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

5. Set the **Destination** where you want to send your data. Select **default-store**. For details on the different destinations, see Destinations.



6. (Optional) To process the data transferred in the flow, select **Add New Processing Step**. For details, see Processing steps. For example:

    1. Add a **Classify**, a **Parse**, and a **Reduce** step, in that order, to automatically remove redundant and empty fields from your data.
    2. To select which messages are processed by the flow, add a **Select Messages** step, and enter a filter into the **AQL Expression** field. For example, to select only the messages received from Fortinet FortiGate firewalls, use the `meta.vendor = fortinet AND meta.product = fortigate` query.



7. Select **Add**, then confirm with **Add**. To save the flow without provisioning it to the routers yet, select **Add as draft** instead.

8. The new flow appears in the **Flows** list.

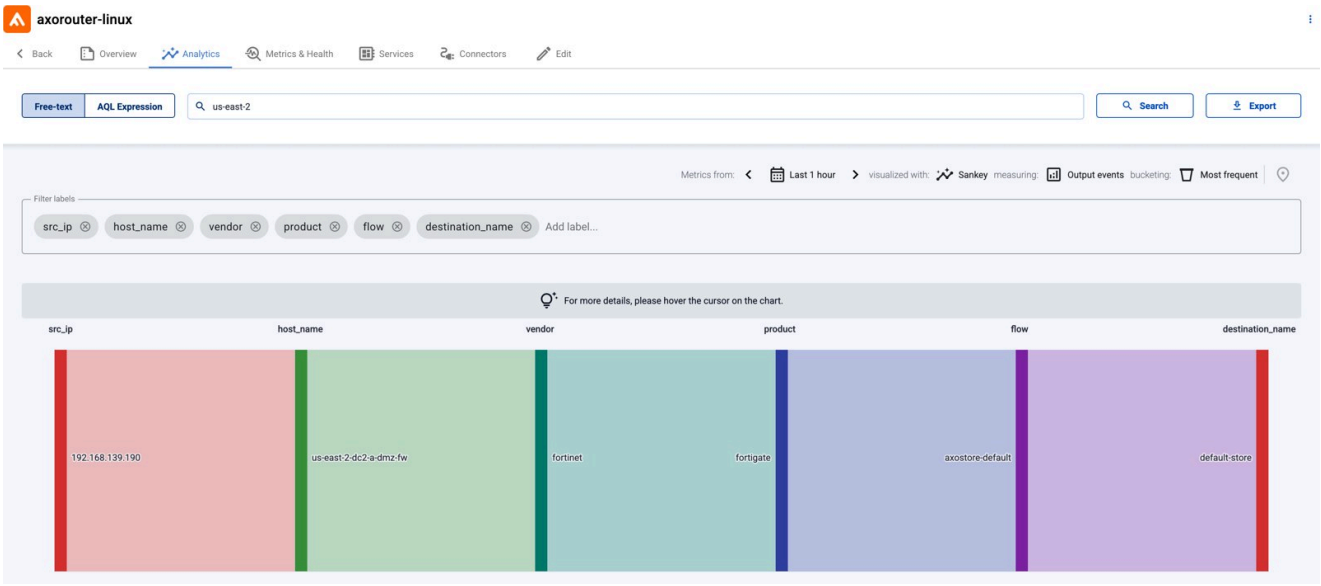


Check the metrics on the Topology page

Open the **Topology** page and verify that your AxoRouter instance is connected both to the source and the destination.

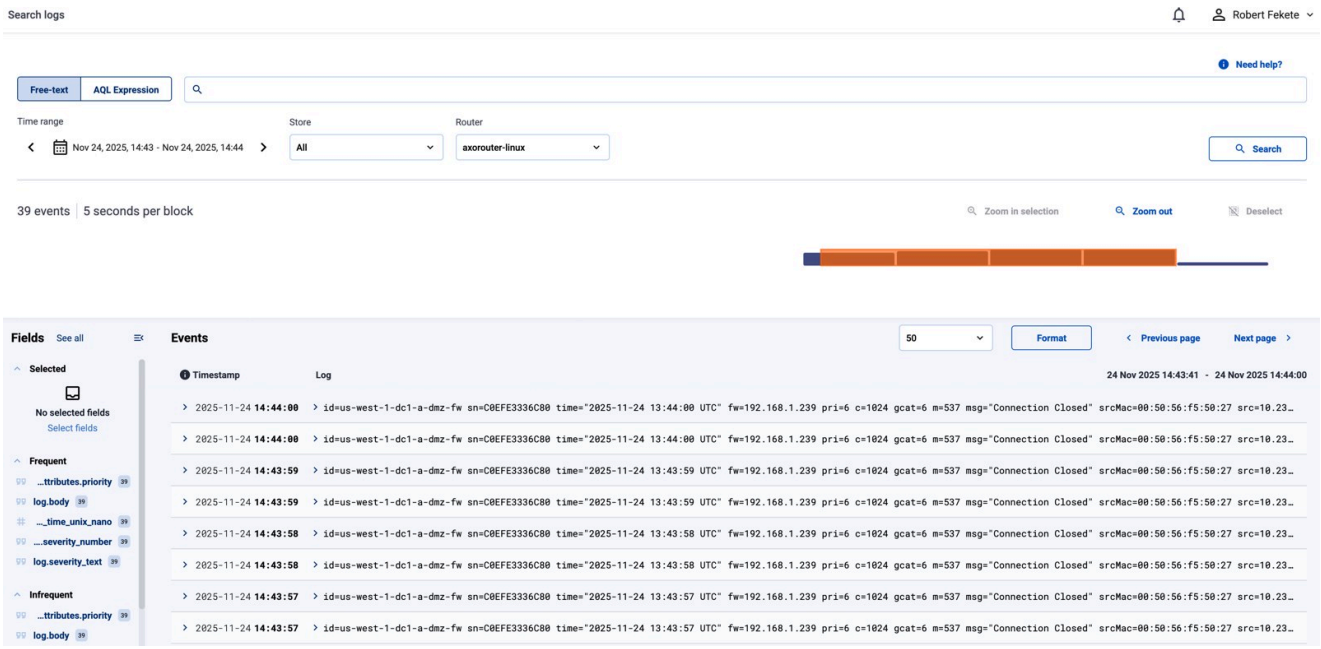


If you have traffic flowing from the source to your AxoRouter instance, the **Topology** page shows the amount of data flowing on the path. Click the AxoRouter instance, then select **Analytics** to visualize the data flow.



Search logs

Open the **Search logs** page, then select **search Search**. The recently received logs in AxoStore will be displayed.



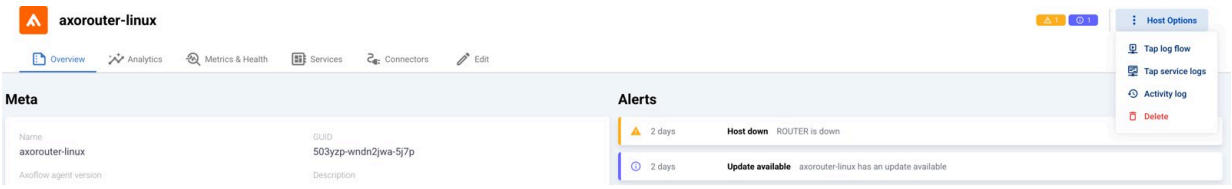
Tap into the log flow

Log tapping in Axoflow samples the log flow. You can use labels to filter for specific messages (like ones with parse errors) and tap only those messages. To not get overwhelmed with events, Axoflow automatically samples the output: if many messages match the selected filter, only a subset is shown (about 1 message per second). Using log tapping, you can quickly troubleshoot both parsing/curation errors and destination ingest (API) errors, and check:

- What was in the original message?
- What is sent in the final payload to the destination?

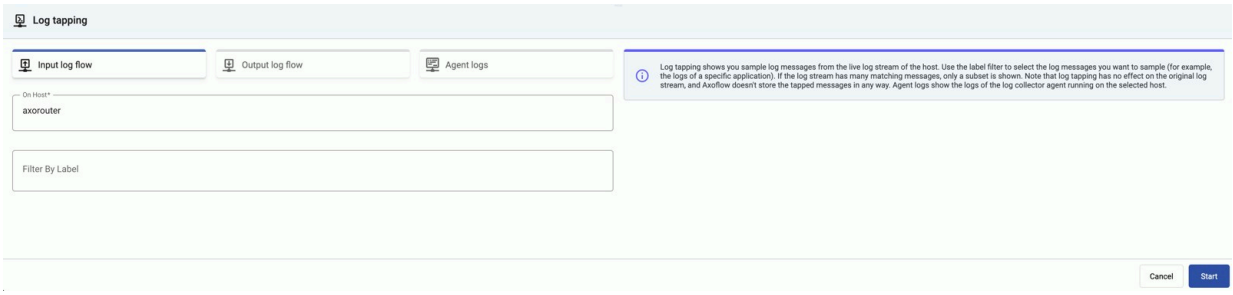
Tap into the log flow.

1. Click your AxoRouter instance on the **Topology** page, then select ⋮ **Host Options** > **Tap log flow**.

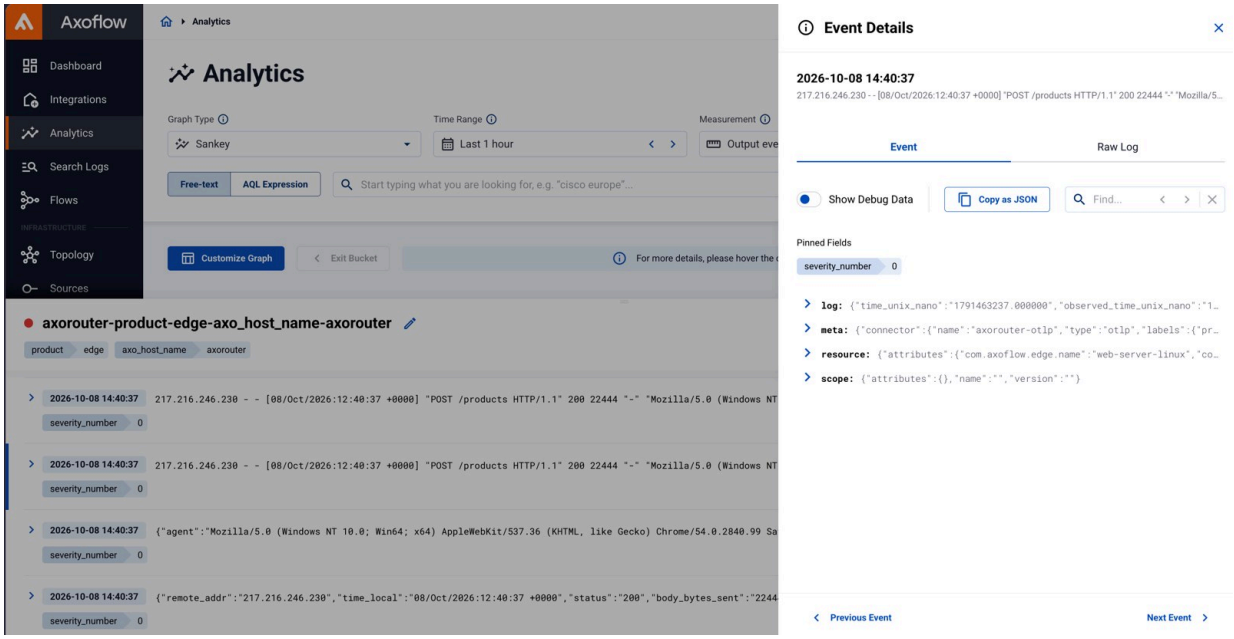


2. Tap into the log flow.
   - To see the input data, select **Input log flow** > **Start**.
   - To see the output data, select **Output log flow** > **Start**.

   You can use labels to filter the messages and sample only the matching ones.



3. When the logs you're interested in show up, click **Pause**, then click a log message to see its details.



Troubleshooting

In case you run into problems, or you're not getting any data in AxoStore, check the logs of your AxoRouter instance:

1. Select **Topology**, then select your AxoRouter instance. Alternatively, select `⌘/Ctrl + K` and enter the name of your AxoRouter.
2. Select ⋮ **Host Options > Tap service logs**.
3. (Optional) To tap only the logs of a specific service, set the **Service name** field.

   Depending on the configuration of the host, the following services can be available:

   - **axoflow-cloudconnector**: The service that collects logs from cloud services and APIs.
   - **axoflow-otel-collector**: The collector agent on edge hosts.
   - **axorouter-logstore**: The service that handles Logstore type stores.
   - **axolet**: The monitoring and management agent for Axoflow pipeline elements.
   - **axorouter-resolver**: A service for retrieving host data from external CMDB systems.
   - **axorouter-syslog**: The main processing element of AxoRouter deployments.
   - **axorouter-wec**: The service that handles the Windows Events connector (WEC).
   - **axostore**: The service that handles AxoStore on the host. Only available if there are stores configured on the host.

4. If systemd is available on the host, select **Systemd service output** to show the logs of the service from the systemd journal. Otherwise, the **Internal logs** of the service are available.

5. Select **Start**. Axoflow displays the log messages of AxoRouter. Check the logs for error messages.

For other problems, see Troubleshooting Axoflow.

3.2 Getting started using a SIEM

Install AxoRouter, connect a source, and send its data to Splunk or another SIEM through your first flow.

This guide shows you how to get started with Axoflow. You're going to install AxoRouter, and configure or create a source to send data to AxoRouter. You'll also configure AxoRouter to forward the received data to your destination SIEM or storage provider. The resulting topology will look something like this:



Why use Axoflow

Using the Axoflow security data pipeline automatically corrects and augments the security data you collect, resulting in high-quality, curated, SIEM-optimized data. It also removes redundant data to reduce storage and SIEM costs. In addition, it allows automates pipeline configuration and provides metrics and alerts for your telemetry data flows.

Prerequisites

You'll need:

- An Axoflow subscription, access to a free evaluation version, or an on-premise deployment.
- A data source. This can be any host that you can configure to send syslog or OpenTelemetry data to your AxoRouter instance that you'll install. If you don't want to change the configuration of an existing device, you can use a virtual machine or a docker container on your local computer.
- A host that you'll install AxoRouter on. This can be a separate Linux host, or a virtual machine running on your local computer.

  AxoRouter supports the following operating systems:

  - Ubuntu 24.04 and 26.04,
  - Red Hat Enterprise Linux 8.8 or newer, 9.4 or newer, and 10.

  Other compatible and derivative distributions (for example, AlmaLinux or Rocky Linux) should work as well, but we don't test them explicitly.

- A destination where AxoRouter will send the received data. This can be a supported SIEM or storage provider, like Splunk or Amazon S3 that you have access to. For a quick test of Axoflow, you can use a free Splunk or OpenObserve account as well. See our list of supported Destinations.

Network access

The host must be able to access the following domains related to the AxoConsole:

- When using AxoConsole SaaS:
  - `<your-tenant-id>.cloud.axoflow.io` : HTTPS traffic on TCP port 443, needed to download the binaries for Axoflow software (like Axolet and AxoRouter).
  - `kcp.<your-tenant-id>.cloud.axoflow.io` : HTTPS (mutual TLS) traffic on TCP port 443 for management traffic.
  - `telemetry.<your-tenant-id>.cloud.axoflow.io` : HTTPS (mutual TLS) traffic on TCP port 443, where Axolet sends the metrics of the host.
  - `us-docker.pkg.dev` : HTTPS traffic on TCP port 443, but only if you're forcing the installation script to pull the container images from the public repository. The Axolet and AxoRouter for Linux installation scripts download the images directly from AxoConsole.
- When using an on-premise AxoConsole:
  - The following domains should point to AxoConsole IP address to access Axoflow from your desktop and AxoRouter hosts:
    - `your-host.your-domain` : The main domain of your AxoConsole deployment.
    - `idp.your-host.your-domain` : A subdomain for the identity provider.
    - `kcp.your-host.your-domain` : A subdomain for management traffic.
    - `telemetry.your-host.your-domain` : A subdomain for agent telemetry.
  - The AxoConsole host must have the following **Open Ports**:
    - 80 TCP: Traefik ingress (HTTP, redirected to HTTPS).
    - 443 TCP: HTTPS for the AxoConsole UI, API, and all web traffic.
- When installing Axoflow agent for Windows or Linux:
  - `github.com` : HTTPS traffic on TCP port 443, for downloading installer packages.

> **Note**
>
> **If you've already completed the Getting started using AxoStore, you can skip to the Add a destination section.**

Log in to the AxoConsole

Verify that you have access to the AxoConsole.

1. Open `https://<your-tenant-id>.axoflow.io/` in your browser.
2. Log in using Google Authentication.

Deploy an AxoRouter instance

Deploy an AxoRouter instance that will route, curate, and enrich your log data.

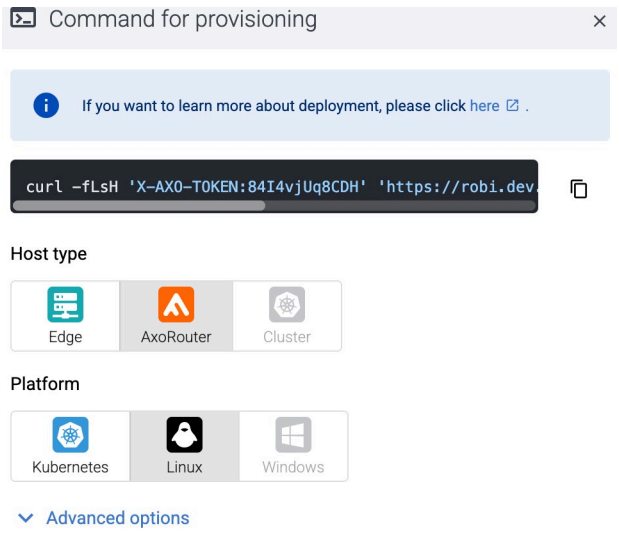
> **Note**
>
> **Note that AxoRouter and Axoflow agent collects detailed, real-time metrics about the data-flows – giving you observability over the health of the security data pipeline and its components. Your security data remains in your self-managed cloud or in your on-prem instance where your sources, destinations, Axoflow agents, and AxoRouters are running, only metrics are forwarded to AxoConsole.**

Deploy AxoRouter on Linux. For other platforms, see AxoRouter.

1. Select **Routers > Add Router**.



2. Select the platform (**Linux**). The one-liner installation command is displayed.



3. (Optional) If you don't want to store any logs locally on AxoRouter, disable AxoStore, select **Advanced options**, scroll down, and deselect **Enable AxoStore**.

4. (Optional)

   If needed, set the **Advanced options** (for example, proxy settings, or **External image pull** in air-gapped environments) to modify the installation parameters. Usually, you don't have to use advanced options unless the Axoflow support team instructs you to do so.

5. Open a terminal on the host where you want to install AxoRouter.

6. Run the one-liner, then follow the on-screen instructions.

   > **Note**
   >
   > **Running the provisioning command with `sudo` would mask environment variables of the calling shell. Either start the whole procedure from a root shell, or let the install script call sudo when it needs to. In other words: don't add the `sudo` command to the provisioning command.**

   Example output:

```
Do you want to install AxoRouter now? [Y]

% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  5480  100  5480    0     0  32076      0 --:--:-- --:--:-- --:--:--
33414
Selecting previously unselected package axorouter.
(Reading database ... 17697 files and directories currently installed.)
Preparing to unpack axorouter.deb ...
Unpacking axorouter (0.66.0) ...
Setting up axorouter (0.66.0) ...
Low maximum socket receive buffer size value detected: 7500000 bytes (7.2MB).
Do you you want to permanently set the net.core.rmem_max sysctl value to
33554432 bytes (32MB) on this system? [Y]

net.core.rmem_max = 33554432
Created symlink '/etc/systemd/system/multi-user.target.wants/axostore.path'
→ '/etc/systemd/system/axostore.path'.
Created symlink '/etc/systemd/system/multi-user.target.wants/axorouter-
wec.path' → '/etc/systemd/system/axorouter-wec.path'.
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 42.9M  100 42.9M    0     0  28.1M      0  0:00:01  0:00:01 --:--:--
28.2M
Selecting previously unselected package axolet.
(Reading database ... 17707 files and directories currently installed.)
Preparing to unpack axolet.deb ...
Unpacking axolet (0.66.0) ...
Setting up axolet (0.66.0) ...
Created symlink '/etc/systemd/system/multi-user.target.wants/axolet.service'
→ '/usr/lib/systemd/system/axolet.service'.
Now continue with onboarding the host on the Axoflow web UI.
```

Note

If you get the following errors:

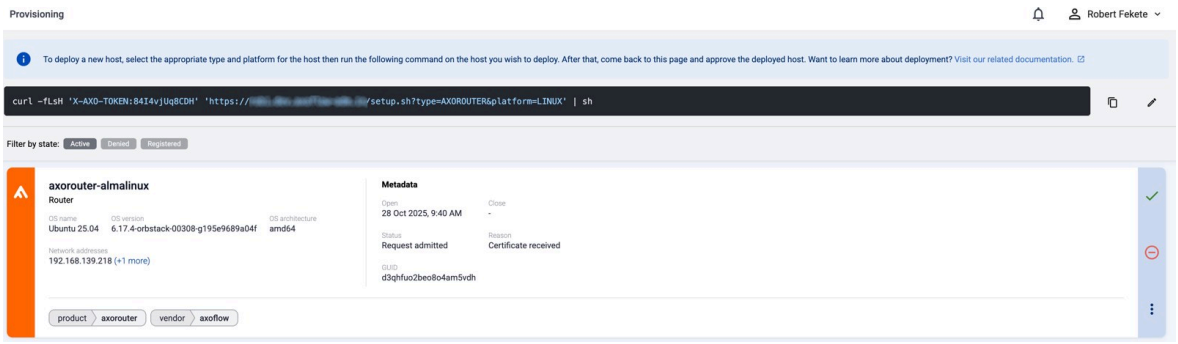
```
AXOROUTER_FUSE envvar is not set. It may mean that the
/usr/local/bin/setup-axorouter script is called without proper
configuration.
Failed to configure AxoRouter.
Failed to deploy the AxoRouter.
```

Set the following (or equivalent) in the `sudoers` file on the host, and rerun the installation command.

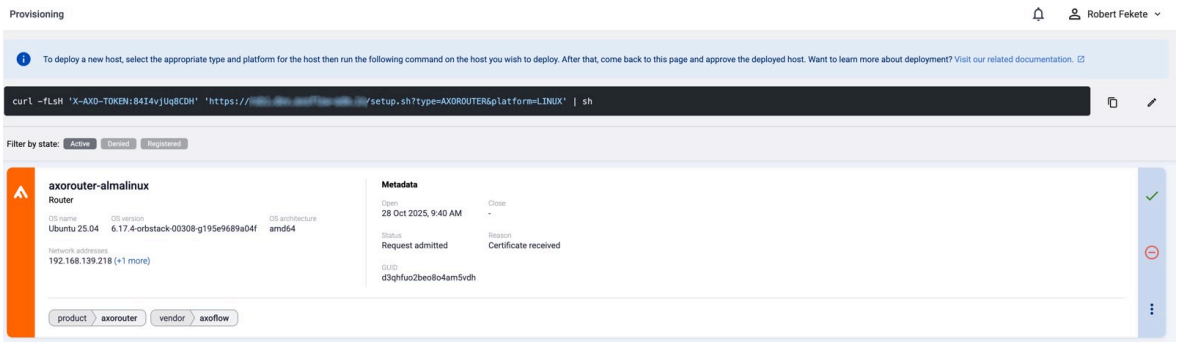
```
Defaults!/usr/local/bin/setup-axorouter env_keep += "AXO* http_proxy
https_proxy no_proxy"
Defaults!/usr/local/bin/axorouter-ctl env_keep += "AXO* http_proxy
https_proxy no_proxy"
Defaults!/usr/local/bin/configure-axolet env_keep += "AXO* http_proxy
https_proxy no_proxy"
```

7. Register the host.

    1. Reload the **Provisioning** page. There should be a registration request for the new AxoRouter deployment. Select ✓.



    2. Select **Register** to register the host. You can add a description and labels (in `label:value` format) to the host.



3. If the primary IP address (the first IP address shown in the **Network addresses** section on the **Routers** page for each AxoRouter) is not accessible from your edge hosts, set a **Network address override** (IP address or an FQDN) that's accessible. Otherwise, data forwarding from edge hosts will fail.

4. Select the **Topology** page. The new AxoRouter instance is displayed.

Add a source

Configure a host to send data to AxoRouter.

Configure a generic syslog host. For sources that are specifically supported by Axoflow, see Sources.

1. Log in to your device. You need administrator privileges to perform the configuration.

2. If needed, enable syslog forwarding on the device.

3. Set AxoRouter as the syslog server. Typically, you can configure the following parameters:

   - **Name or IP Address of the syslog server**: Set the address of your AxoRouter.
   - **Protocol**: If possible, set TCP or TLS.

     > **Note**
     >
     > **If you're sending data over TLS, make sure to configure a TLS-enabled connector rule in Axoflow.**

   - **Syslog Format**: If possible, set RFC5424 (or equivalent), otherwise leave the default.
   - **Port**: Set a port appropriate for the protocol and syslog format you have configured.

By default, AxoRouter accepts data on the following ports (unless you've modified the default connector rules):

- 514 UDP and TCP for RFC3164 (BSD-syslog) and RFC5424 (IETF-syslog) formatted traffic. AxoRouter automatically recognizes and handles both formats.
- 601 TCP (RFC 3195 - Reliable Delivery for syslog) for RFC5424 (IETF-syslog) and RFC3164 (BSD-syslog) formatted traffic. AxoRouter automatically recognizes and handles both formats.
- 6514 TCP for TLS-encrypted syslog traffic (RFC 5425).
- 4317 TCP for OpenTelemetry log data.
- 5085 TCP for the Windows Event Collector (WEC) service.
- 8088 TCP for the Splunk HEC receiver (optional, only if you forward Splunk HEC traffic to AxoRouter).
- 9200 TCP for the Elasticsearch Bulk API connector (optional, only if you forward data from Elastic agents to AxoRouter).

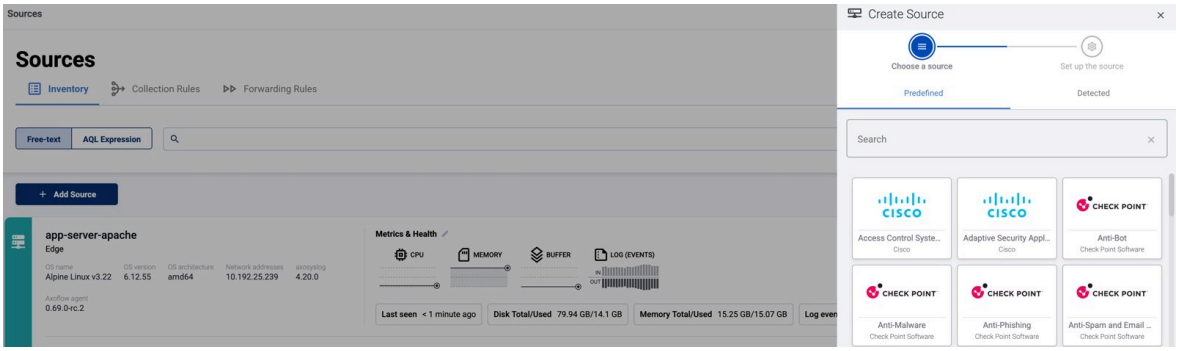
To receive data on other ports or other protocols, configure other connector rules for the AxoRouter host.

For TLS-encrypted syslog connections, create a new connector rule or edit an existing one, and configure the keys and certificates needed to encrypt the connections. For details, see Syslog.

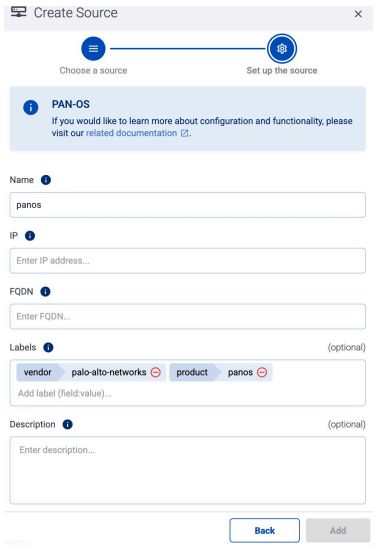
> **Note**
>
> **When you create a connector rule on a new port, open that port in the firewall of of every corresponding AxoRouter as well.**

4. Add the source to AxoConsole.

    1. Open the AxoConsole and select **Topology**.
    2. Select **Add Item > Source**.



- If the source is actively sending data to an AxoRouter instance, select **Detected**, then select your source(s) and select **Add Selected Source(s)**.
- Otherwise, select the vendor and product corresponding to your source from the **Predefined** sources, then enter the parameters of the source, like **IP address** and **FQDN**.



> ## Note
>
> **During log tapping, you can add hosts that are actively sending data to an AxoRouter instance by clicking Register source.**

3. (Optional) Add custom labels as needed.

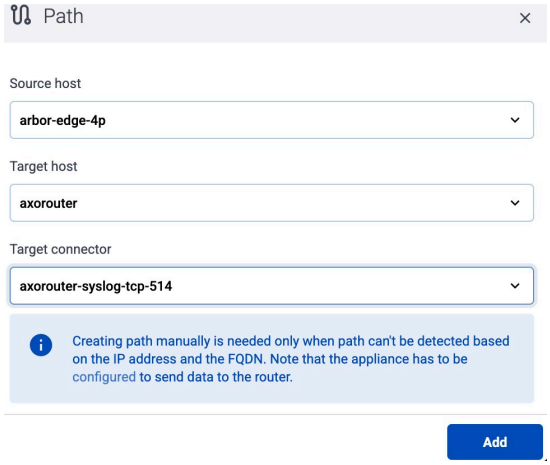
> **CAUTION:**
>
> AxoConsole can automatically raise an alert if a source that used to send data regularly stops sending data. To enable this alert for a source, add the `traffic_alerts:true` label to the source.

4. Select **Add**.

5. (Optional) **Add Path** manually. That's needed only when AxoConsole can't detect the path based on the IP address and the FQDN, and you haven't yet configured the source to send data to the router. If you've added the source from the **Detected** list, you can skip this step, as the path is created automatically.

   > **Create a path between the source and the AxoRouter instance.**
   >
   > If you've added the source from the **Detected** list, you can skip this step, as the path is created automatically.

1. Select **Topology > Add Item > Path**.



2. If not set automatically, select the beginning of the path in the **Source host** field.
3. Select the target router or aggregator this source is sending its data to in the **Target host** field, for example, `axorouter`.
4. Select the **Target connector**. The connector determines how the destination receives the data (for example, using which protocol or port).
5. Select **Add**. The new path appears on the **Topology** page.



Note

If your syslog source is running `syslog-ng`, Splunk Connect for Syslog (SC4S), or AxoSyslog as its log forwarder agent, consider installing Axolet on the host and instrumenting the configuration of the log forwarder to receive detailed metrics about the host and the processed data. For details, see Manage and monitor the pipeline.

Add a destination

Add the destination where you're sending your data. For a quick test, you can use a free Splunk or OpenObserve account.

Add a Splunk Cloud destination. For other destinations, see Destinations.

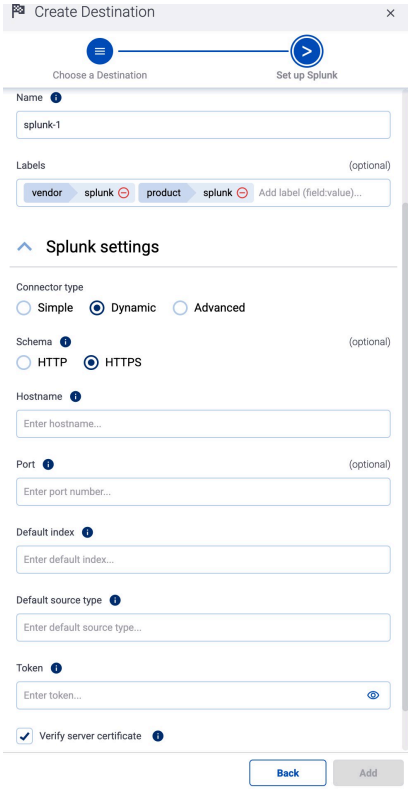
Prerequisites

1. Enable the HTTP Event Collector (HEC) on your Splunk deployment if needed. On Splunk Cloud Platform deployments, HEC is enabled by default.
2. Create a token for Axoflow to use in the destination. When creating the token, use the syslog source type.

   For details, see Set up and use HTTP Event Collector in Splunk Web.
3. If you're using AxoRouter, create the indexes where Axoflow sends the log data. Which index is needed depends on the sources you have, but create at least the following event indices: `axoflow`, `infraops`, `netops`, `netfw`, `osnix` (for unclassified messages). Check your sources in the Sources section for a detailed lists on which indices their data is sent.
4. If you've created any new indexes, make sure to add those indexes to the token's Allowed Indexes.

Steps

1. Create a new destination.
   1. Open the AxoConsole.
   2. Select **Destinations > Add Destination > Destination**.
2. Select **Splunk**.
3. Select **Dynamic**. This will allow you to set a default index, source, and source type for messages that aren't automatically identified.



4. Enter your Splunk URL into the **Hostname** field, for example, `<your-splunk-tenant-id>.splunkcloud.com` for Splunk Cloud Platform free trials, or `<your-splunk-tenant-id>.splunkcloud.com` for Splunk Cloud Platform instances.

5. Enter the name of the **Default Index**. The data will be sent into this index if no other index is set during the processing of the message (based on automatic classification, or by the processing steps of the **Flow**). Make sure that the index exists in Splunk.

6. Enter the **Default Source Type**. This will be assigned to the messages that have no sourcetype set during the processing of the message (based on automatic classification, or by the processing steps of the **Flow**).

7. Enter the token you've created into the **Token** field.

8. Disable the **Verify server certificate** option unless your deployment has a valid, non-self-signed certificate. Free Splunk Cloud accounts have self-signed certificates.

9. (Optional) You can set other options as needed for your environment. For details, see Splunk.

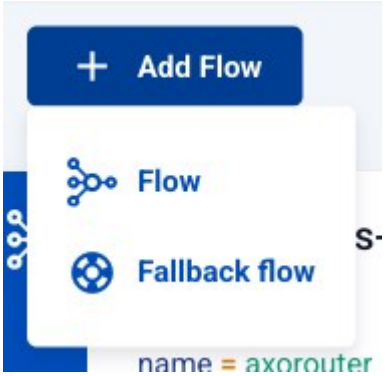
10. Select **Add**.

Create a flow

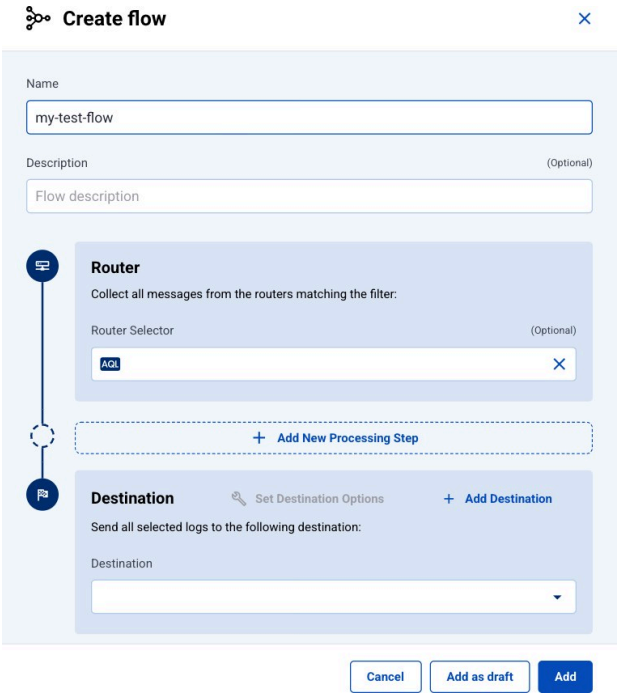
Create a flow to route the traffic from your AxoRouter instance to your destination.

1. Select **Flows**.
2. Select **Add Flow > Flow**.

   To create a fallback flow, select **Add Flow > Fallback flow**.



3. Enter a name for the flow, for example, `my-test-flow`.



4. In the **Router Selector** field, enter an expression that matches the routers you want to apply the flow to. To select a specific router, use a name selector, for example, `name = my-axorouter-hostname`.

   You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

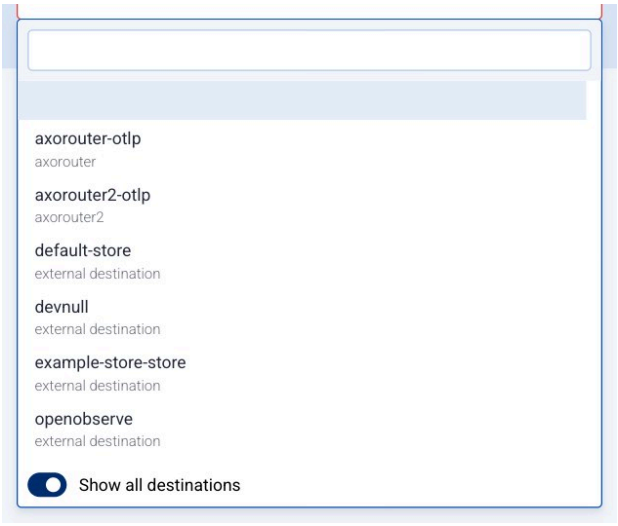
   - If you leave the router selector empty, it matches every AxoRouter instance.

- To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.
- If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

> **Note**
>
> **You can configure multiple fallback flows, but only one fallback flow can apply to an AxoRouter (so the Router Selector of the fallback flows can't overlap).**

5. Select the **Destination** where you want to send your data. If you don't have any destination configured, you can select **Add Destination** in the destination section to create a new destination now. For details on the different destinations, see Destinations.

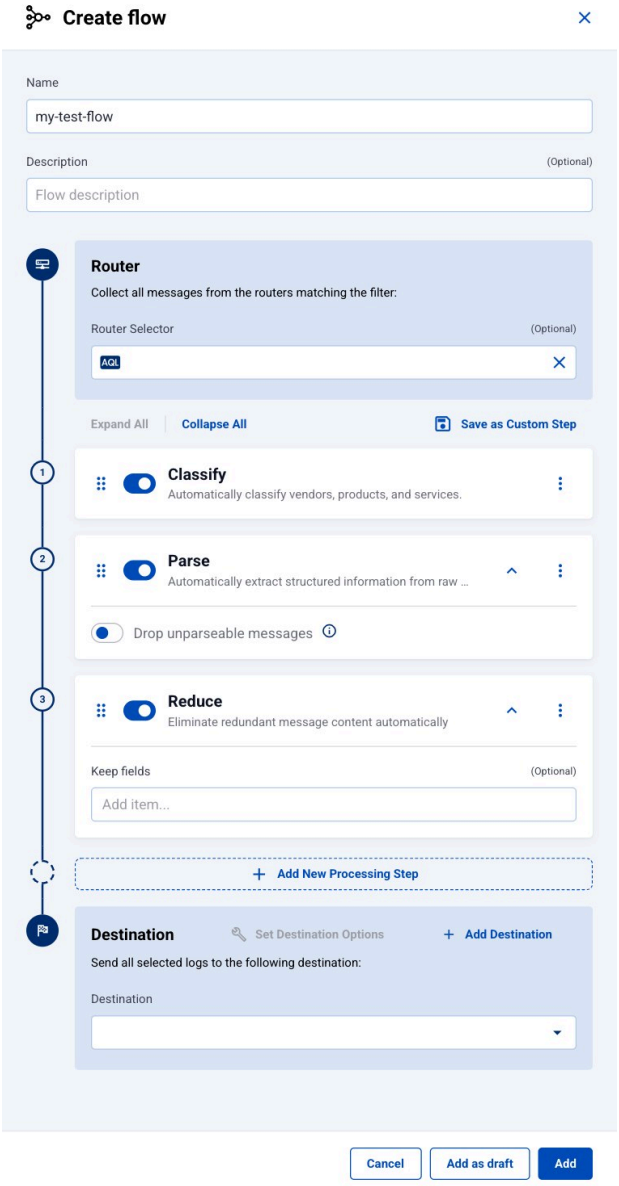
   - If you don't have any destination configured, see Destinations.
   - If you've already created a store, it's automatically available as a destination. Note that the **Router Selector** of the flow must match only AxoRouters that have the selected store available, otherwise you'll get an error message.
   - If you want to send data to another AxoRouter, enable the **Show all destinations** option, and select the connector of the AxoRouter where you want to send the data.



6. (Optional) To process the data transferred in the flow, select **Add New Processing Step**. For details, see Processing steps. For example:

   1. Add a **Classify**, a **Parse**, and a **Reduce** step, in that order, to automatically remove redundant and empty fields from your data.

2. To select which messages are processed by the flow, add a **Select Messages** step, and enter a filter into the **AQL Expression** field. For example, to select only the messages received from Fortinet FortiGate firewalls, use the `meta.vendor = fortinet AND meta.product = fortigate` query.



7. Select **Add**, then confirm with **Add**. To save the flow without provisioning it to the routers yet, select **Add as draft** instead.

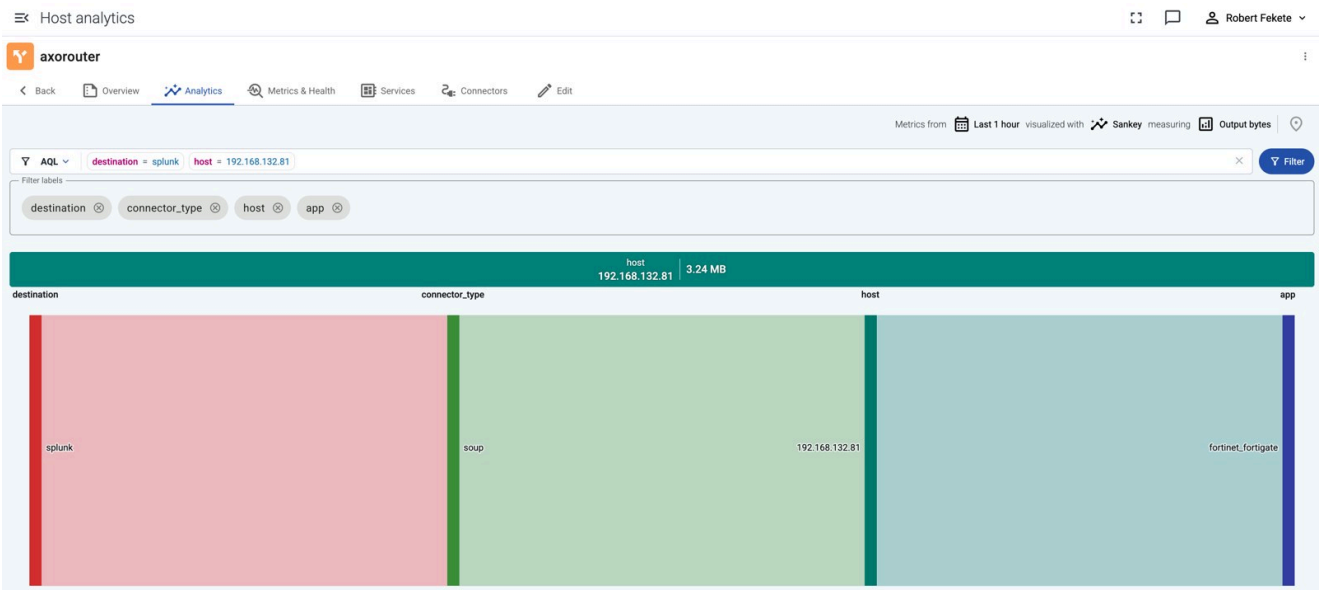
8. The new flow appears in the **Flows** list.



Check metrics on the Topology page

Open the **Topology** page and verify that your AxoRouter instance is connected both to the source and the destination.

If you have traffic flowing from the source to your AxoRouter instance, the **Topology** page shows the amount of data flowing on the path. Click the AxoRouter instance, then select **Analytics** to visualize the data flow.



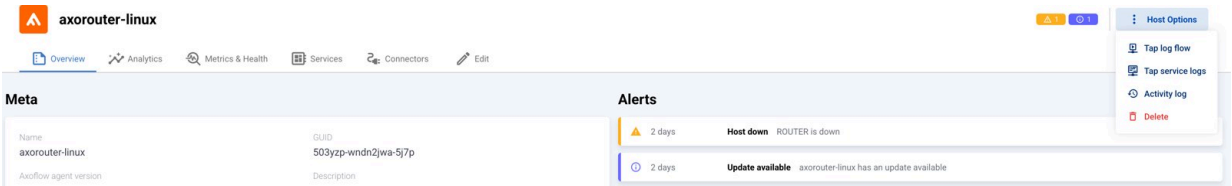
Tap into the log flow

Log tapping in Axoflow samples the log flow. You can use labels to filter for specific messages (like ones with parse errors) and tap only those messages. To not get overwhelmed with events, Axoflow automatically samples the output: if many messages match the selected filter, only a subset is shown (about 1 message per second). Using log tapping, you can quickly troubleshoot both parsing/curation errors and destination ingest (API) errors, and check:

- What was in the original message?
- What is sent in the final payload to the destination?

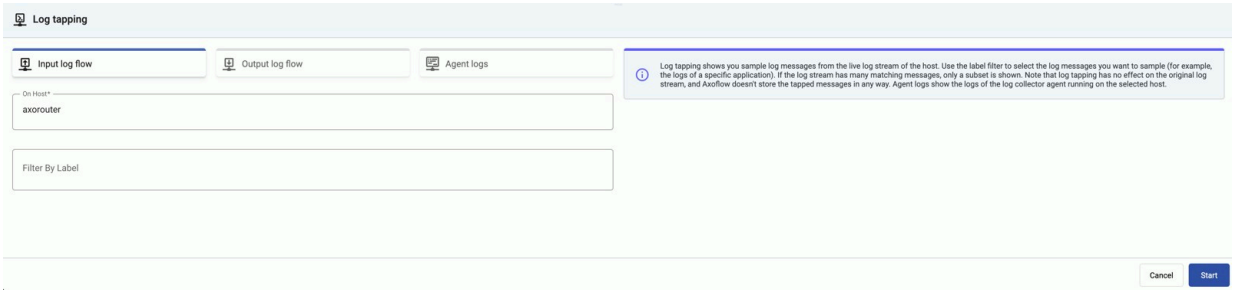
Tap into the log flow.

1. Click your AxoRouter instance on the **Topology** page, then select ⋮ **Host Options** > **Tap log flow**.

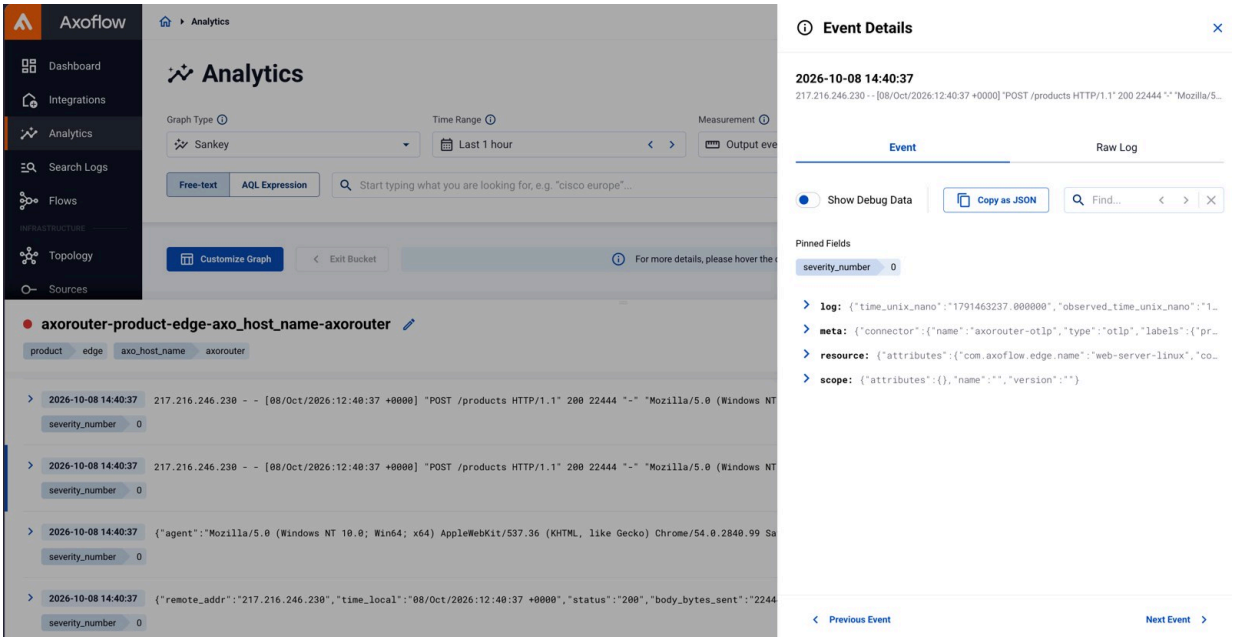


2. Tap into the log flow.
   - To see the input data, select **Input log flow > Start**.
   - To see the output data, select **Output log flow > Start**.

   You can use labels to filter the messages and sample only the matching ones.



3. When the logs you're interested in show up, click **Pause**, then click a log message to see its details.



Troubleshooting

In case you run into problems, or you're not getting any data in Splunk, check the logs of your AxoRouter instance:

1. Select **Topology**, then select your AxoRouter instance. Alternatively, select `⌘/Ctrl + K` and enter the name of your AxoRouter.
2. Select ⋮ **Host Options > Tap service logs**.
3. (Optional) To tap only the logs of a specific service, set the **Service name** field.

   Depending on the configuration of the host, the following services can be available:

- **axoflow-cloudconnector**: The service that collects logs from cloud services and APIs.
- **axoflow-otel-collector**: The collector agent on edge hosts.
- **axorouter-logstore**: The service that handles Logstore type stores.
- **axolet**: The monitoring and management agent for Axoflow pipeline elements.
- **axorouter-resolver**: A service for retrieving host data from external CMDB systems.
- **axorouter-syslog**: The main processing element of AxoRouter deployments.
- **axorouter-wec**: The service that handles the Windows Events connector (WEC).
- **axostore**: The service that handles AxoStore on the host. Only available if there are stores configured on the host.

4. If systemd is available on the host, select **Systemd service output** to show the logs of the service from the systemd journal. Otherwise, the **Internal logs** of the service are available.

5. Select **Start**. Axoflow displays the log messages of AxoRouter. Check the logs for error messages.

Some common errors include:

- `Redirected event for unconfigured/disabled/deleted index=netops with source="source::axo" host="host::axosyslog-almalinux" sourcetype="sourcetype::fortigate_event" into the LastChanceIndex. So far received events from 1 missing index(es).` : The Splunk index where AxoRouter is trying to send data doesn't exist. Check which index is missing in the error message and create it in Splunk. (For a list of recommended indices, see the Splunk destination prerequisites.)
- `http: error sending HTTP request; url='https://prd-p-sp2id.splunkcloud.com:8088/services/collector/event/1.0?index=&source=&sourcetype=', error='SSL peer certificate or SSH remote key was not OK', worker_index='0', driver='splunk--flow-axorouter4-almalinux#0', location='/usr/share/syslog-ng/include/scl/splunk/splunk.conf:104:3'` : Your Splunk deployment uses an invalid or self-signed certificate, and the **Verify server certificate** option is enabled in the Splunk destination of Axoflow. Either fix the certificate in Splunk, or: select **Topology > <your-splunk-destination>**, disable **Verify server certificate**, then select **Update**.

For other problems, see Troubleshooting Axoflow.

4 Deployment scenarios

Sample architectures for inserting AxoRouter into your existing data pipeline, and for deploying Axoflow agents or onboarding your existing log collectors.

Thanks to its flexible deployment modes, you can quickly insert an Axoflow and its processing node (called AxoRouter) transparently into your data pipeline and gain instant benefits:

- Data reduction
- Improved SIEM accuracy
- Configuration UI for routing data
- Automatic data classification
- Metrics about log ingestion, processing, and data drops
- Analytics about the transported data
- Health check, highlighting anomalies

> **Note**
>
> **Note that AxoRouter and Axoflow agent collects detailed, real-time metrics about the data-flows – giving you observability over the health of the security data pipeline and its components. Your security data remains in your self-managed cloud or in your on-prem instance where your sources, destinations, Axoflow agents, and AxoRouters are running, only metrics are forwarded to AxoConsole.**

After the first step, you can further integrate your pipeline, by deploying Axoflow agents to collect and manage your security data, or onboarding your existing log collector agents (for example, syslog-ng). Let's see what these scenarios look like in detail.

4.1 AxoConsole deployment

Use AxoConsole as a SaaS service, or deploy it on premises in your own environment.

You can use AxoConsole:

- as a SaaS solution, without any deployment (request a free trial),
- deploy it on-premises, or
- deploy it in air-gapped environments (contact us for details).

We also support hybrid environments.

The AxoConsole provides the UI for accessing metrics and analytics, deploying and configuring AxoRouter instances, configuring data flows, and so on.

4.2 Transparent router mode

Deploy AxoRouter in front of your SIEM, and point your sources to it to get classification, parsing, enrichment, and metrics.

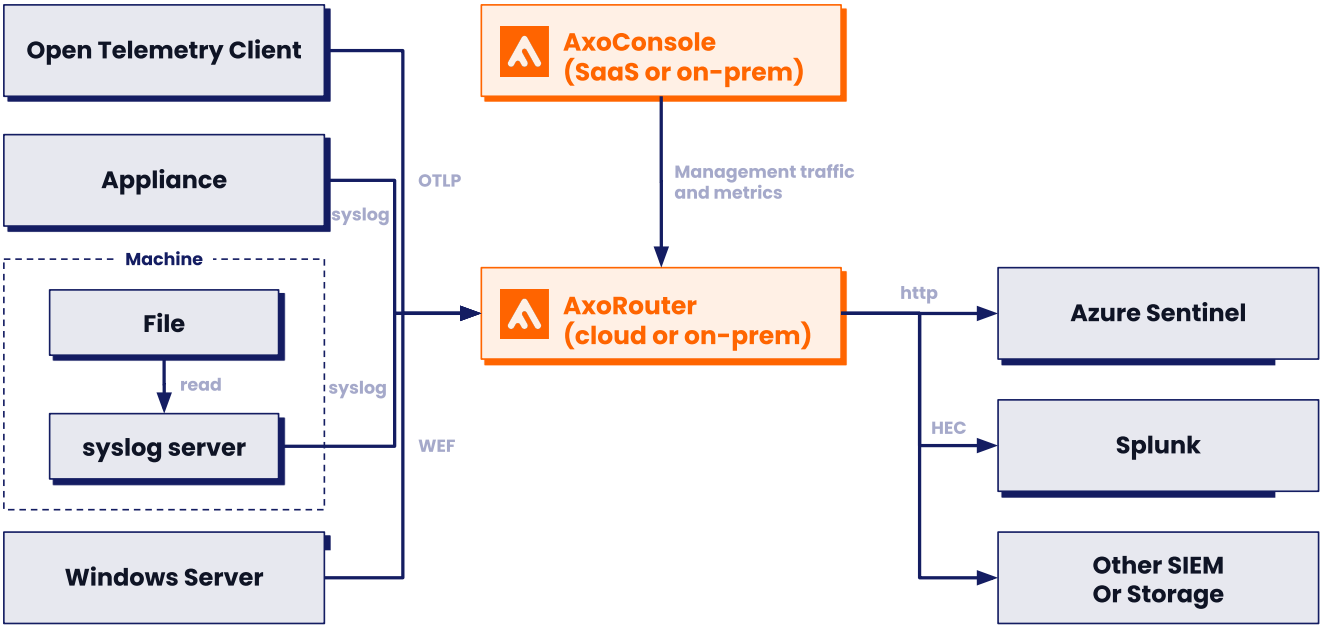
AxoRouter is a powerful aggregator and data processing engine that can receive data from a wide variety of sources, including:

- OpenTelemetry,
- syslog,
- Windows Event Forwarding, or HTTP.

In transparent mode, you deploy AxoRouter in front of your SIEM and configure your sources or aggregators to send the logs to AxoRouter instead of the SIEM. The transparent deployment method is a quick, minimally invasive way to get instant benefits and value from Axoflow, and:

- automatically identify the sending device or application,
- apply device-specific, zero-maintenance parsers to the incoming data,
- enrich the incoming messages with metadata, and
- get metrics about the data throughput.

> **Note**
>
> **Note that AxoRouter and Axoflow agent collects detailed, real-time metrics about the data-flows – giving you observability over the health of the security data pipeline and its components. Your security data remains in your self-managed cloud or in your on-prem instance where your sources, destinations, Axoflow agents, and AxoRouters are running, only metrics are forwarded to AxoConsole.**


Open Telemetry Client
Appliance
Machine
File
read
syslog server
Windows Server
OTLP
syslog
syslog
WEF
AxoConsole
(SaaS or on-prem)
Management traffic
and metrics
AxoRouter
(cloud or on-prem)
http
HEC
Azure Sentinel
Splunk
Other SIEM
Or Storage

4.3 Router and edge deployment

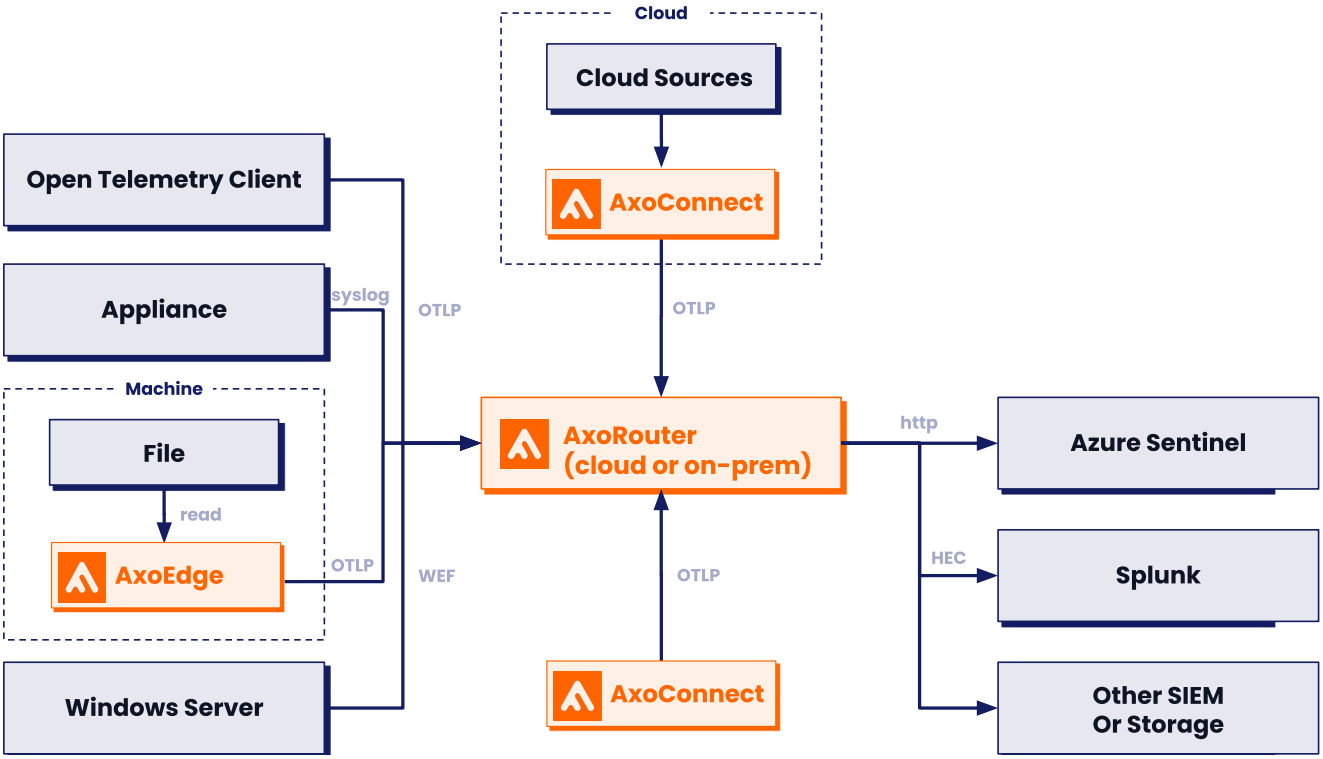
Collect data with Axoflow agents on Kubernetes, Linux, and Windows hosts, and send it to AxoRouter for processing.

Axoflow provides agents to collect data from all kinds of sources:

- Kubernetes clusters,
- cloud sources,
- security appliances,
- Linux servers,
- Microsoft Windows hosts.

(If you'd prefer to keep using your existing syslog infrastructure instead of the Axoflow agents, see Onboard existing syslog/syslog-ng infrastructure).

You can deploy the AxoRouter data aggregator on Linux and Kubernetes.



Using the Axoflow collector agents gives you:

- Reliable transport: Between its components, Axoflow transports security data using the reliable OpenTelemetry protocol (OTLP) for high performance, and to avoid losing messages.
- Managed components: You can configure, manage, monitor, and troubleshoot all these components from the AxoConsole. For example, you can sample the data flowing through each component.

- Metrics: Detailed metrics from every collector provide unprecedented insight into the status of your data pipeline.

4.4 AxoRouter high availability

Deploy multiple AxoRouter instances in a high availability cluster, either behind a load balancer or sharing a virtual IP managed by keepalived.

You can deploy multiple AxoRouter instances as a high availability cluster so the data pipeline keeps accepting logs even when an AxoRouter host fails. There are two main approaches: a load balancer in front of the cluster, or a shared virtual IP managed by `keepalived`.

Load balancer in front of the cluster

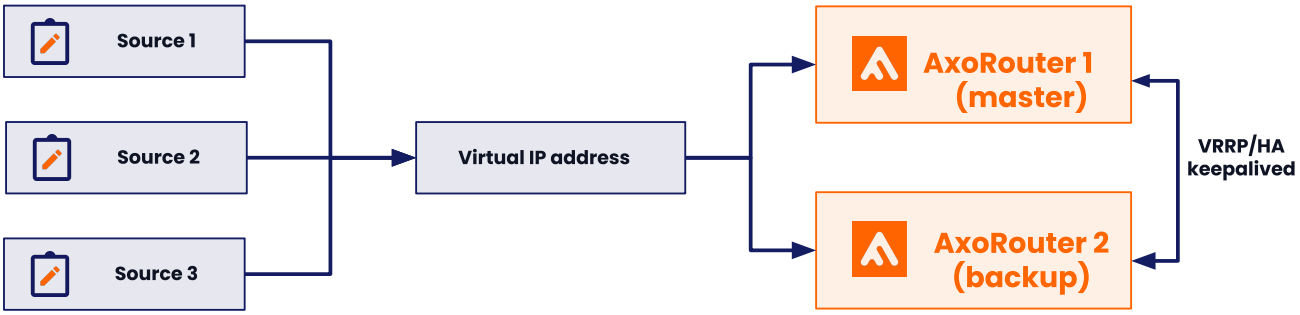
Sources connect to a load balancer, which forwards traffic to one of the AxoRouter hosts. If a host fails, the load balancer routes traffic to the remaining healthy hosts.



This approach reuses an existing load balancer if you already have one, but requires the load balancer to support all relevant protocols (for example, syslog over TCP or TLS).

Virtual IP managed by keepalived

Sources connect to a virtual IP address that floats across the AxoRouter hosts. `keepalived` uses VRRP to elect a primary node, which holds the virtual IP. If the primary node fails, `keepalived` moves the virtual IP to a backup node, and the sources keep reaching AxoRouter without changing their destination address.



This approach doesn't depend on external load balancer infrastructure and suits on-prem deployments well.

Cluster sizing and topology

The hosts running AxoRouter can be physical or virtual machines. The recommended layout depends on which you choose:

- Physical machines: Install two machines with a dedicated HA link between them to keep the HA traffic separate from the production traffic.
- Virtual machines: Set up three VMs with a dedicated (and guaranteed bandwidth) HA link (HA VLAN) spanning across them, to keep the HA traffic separate from the production traffic.

> **Note**
>
> A cluster targeting highly reliable services should ideally consist of at least three nodes to avoid split-brain situations. In a split-brain situation, both nodes think that they are the last surviving node (for example, during a network failure) and bring up their services, leading to several problems.
>
> The setups described here use fewer than the ideal number of nodes for cost reasons. Dedicated HA links provide an additional channel for HA communication independently of the production traffic, and significantly reduce the risk of split-brain scenarios.

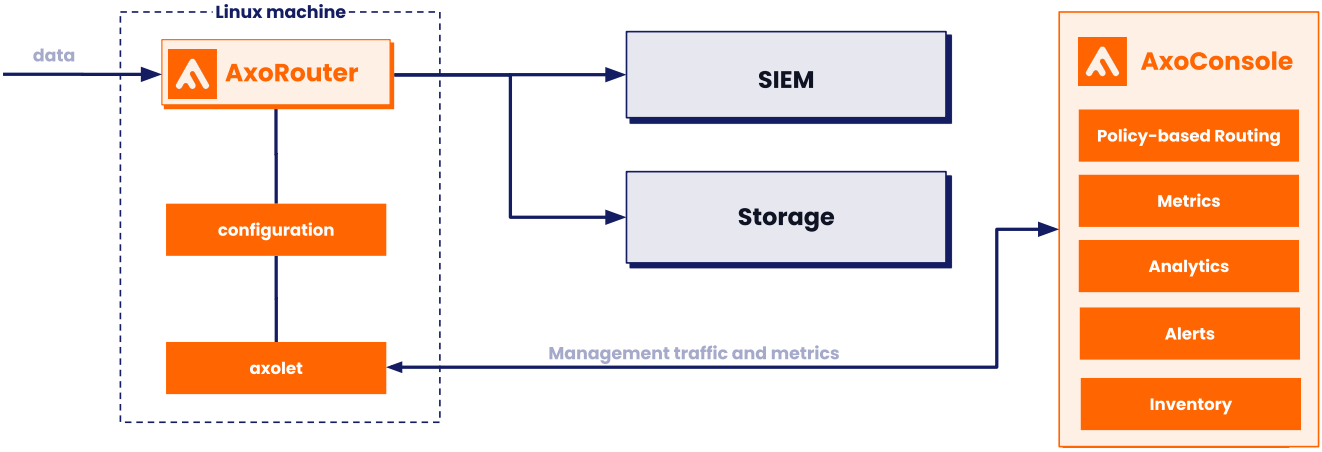
For step-by-step instructions to deploy a `keepalived`-based cluster with a VRRP-managed virtual IP, see Set up AxoRouter high availability with keepalived.

4.5 Onboard existing syslog/syslog-ng infrastructure

Bring an existing syslog or syslog-ng architecture into Axoflow with managed or unmanaged AxoRouter instances, or in read-only mode.

If your organization already has a syslog architecture in place, Axoflow provides ways to reuse it. This allows you to integrate your existing infrastructure with Axoflow, and optionally – in a later phase – replace your log collectors with the agents provided by Axoflow.

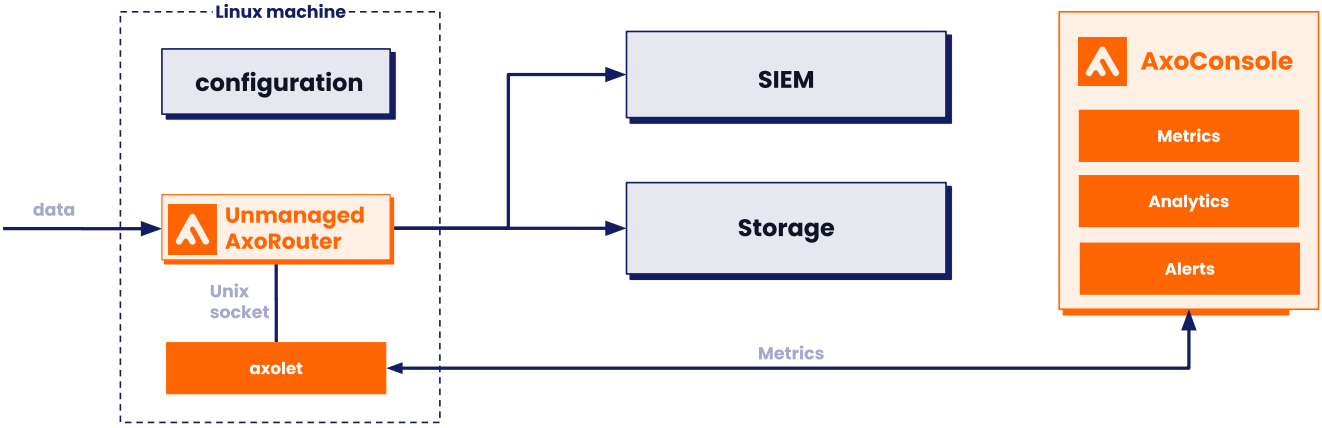
Managed AxoRouter deployments



In this deployment mode you use the centralized management UI of AxoConsole to manage your AxoRouter instances. This provides the tightest integration and the most benefits, including:

- Configuration management from the UI
- Automatic host inventory and host attribution
- Automatic classification and enrichment of incoming data
- Advanced routing based on labels
- Advanced and more detailed metrics about the log ingestion, processing, data drops, delays
- Detailed analytics about the transported data
- Access to the FilterX data processing engine
- Ability to receive OpenTelemetry data
- Acts as a Windows Event Collector server, allowing you to collect Windows events
- Optimized and normalized output for the specific SIEMs
- Data reduction
- Get notifications about alerts and anomalies

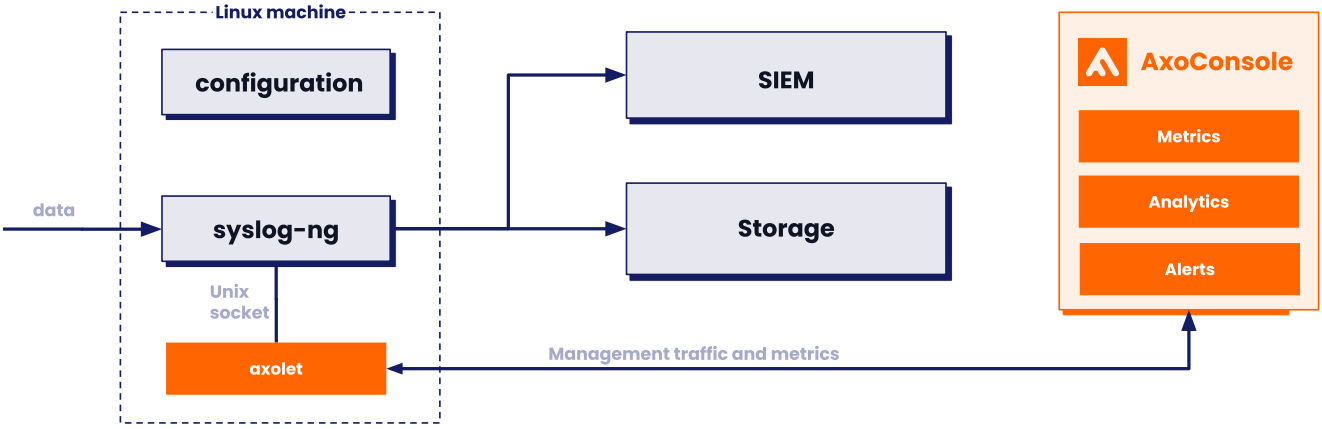
Unmanaged AxoRouter deployments



In this mode, you install AxoRouter on the data source to replace its local collector agent, and manage it manually. That way you get the functional benefits of using AxoRouter as an aggregator and data curation engine to collect and classify your data, but can manage its configuration as you see fit. This gives you all the benefits of the read-only mode (since AxoRouter includes Axolet as well), and in addition, it provides:

- Detailed metrics about the log ingestion, processing, data drops, delays
- Detailed analytics about the transported data
- Access to the FilterX data processing engine
- Ability to receive OpenTelemetry data
- Optimized output for the specific SIEMs
- Data reduction
- Get notifications about alerts and anomalies

Read-only mode with syslog-ng™



In this scenario, you install Axolet on the data source. Axolet is a monitoring (and management) agent that integrates with the local log collector and sends detailed metrics about the host and its data traffic to the AxoConsole. This allows you to use the

AxoConsole to:

- Get a visual overview of your security data pipeline topology
- Check the metrics about the log ingestion, processing, and data drops
- Browse data analytics and health checks
- Get notifications about alerts and anomalies

Axoflow integrates with existing syslog-ng (AxoSyslog, Splunk Connect for Syslog (SC4S), or syslog-ng) deployments by running the Axolet agent besides the syslog-ng process on the host. The agent gathers metrics from syslog-ng through the control socket. To extend the built-in metrics, the agent can be instrumented with a minimal configuration change to provide additional metrics about the log flow. With some additional instrumentation, syslog-ng can be extended with log tapping functionality as well.

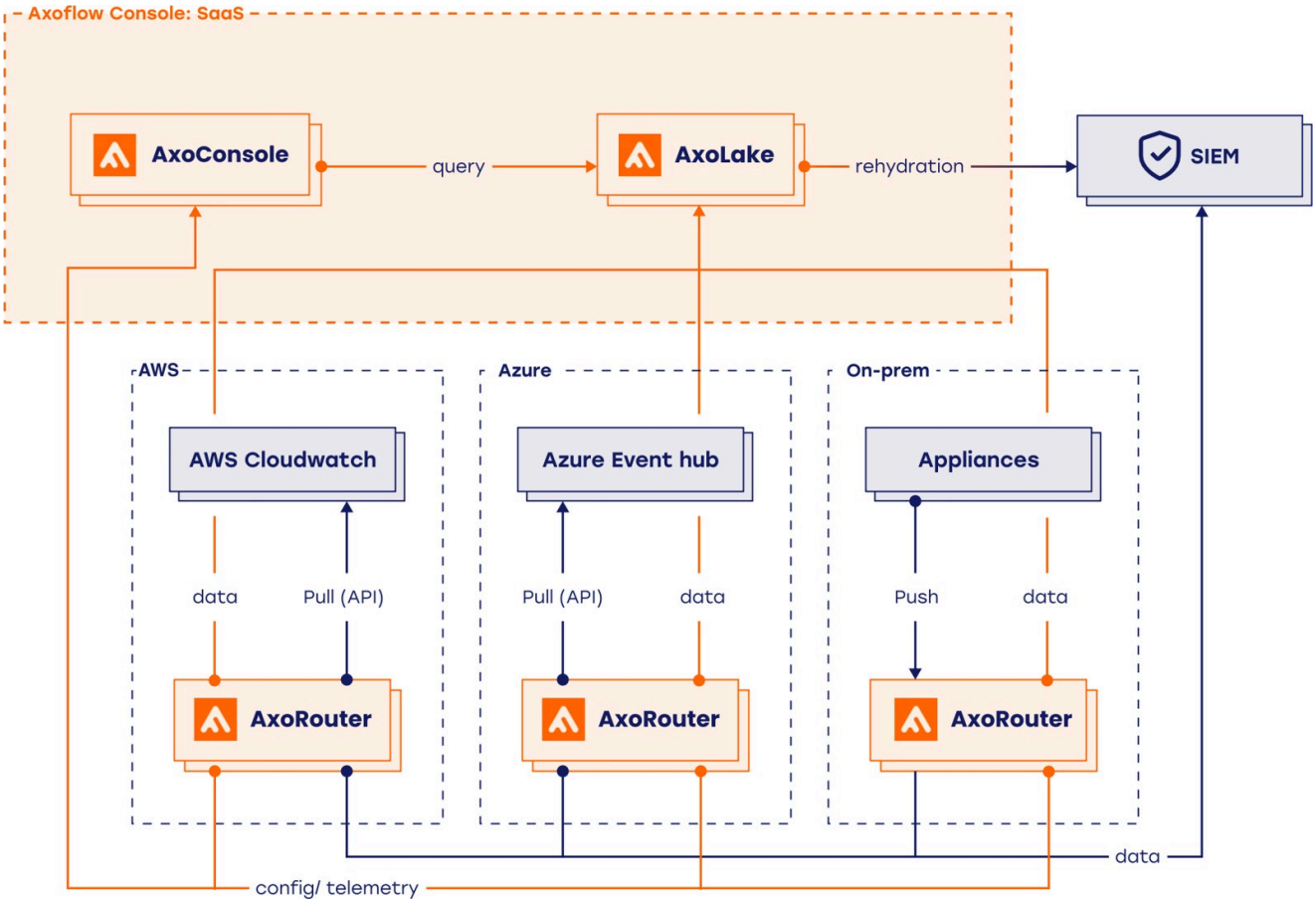
> **Note**
>
> **Axoflow can manage the configuration of AxoRouter and Axoflow agent deployments using AxoConsole. AxoSyslog, SC4S, syslog-ng, or syslog-ng Premium Edition configurations are available centrally in read-only mode. To centrally manage these configurations from AxoConsole upgrade the agents to AxoRouter. (See the Why Enterprises Choose Axoflow Platform vs. syslog-ng™ Premium Edition? comparison page for details.)**

4.6 Multi-cloud and multi-datacenter scenario

Collect security data from several cloud providers and datacenters with local agents and Cloud Connectors, and manage it from one place.

Organizations running workloads across multiple cloud providers (AWS, Azure, GCP) and private datacenters face a common challenge: each environment produces security and observability data in different formats, volumes, and locations, making it hard to collect, normalize, and route it consistently.

Axoflow is designed for exactly this scenario. You deploy Axoflow agents and Axoflow Cloud Connectors close to your data sources in each cloud and datacenter. The agents forward data to one or more AxoRouter nodes, which you can place per region to minimize egress costs, or centralize for simpler operations.



This approach gives you:

- Single management plane: You configure and monitor all Axoflow agents and AxoRouter nodes from the AxoConsole, regardless of which cloud or datacenter they run in.
- Local aggregation: AxoRouters automatically filter, enrich, and reduce data before forwarding it, so you only pay cross-cloud or cross-datacenter egress for the data that matters.

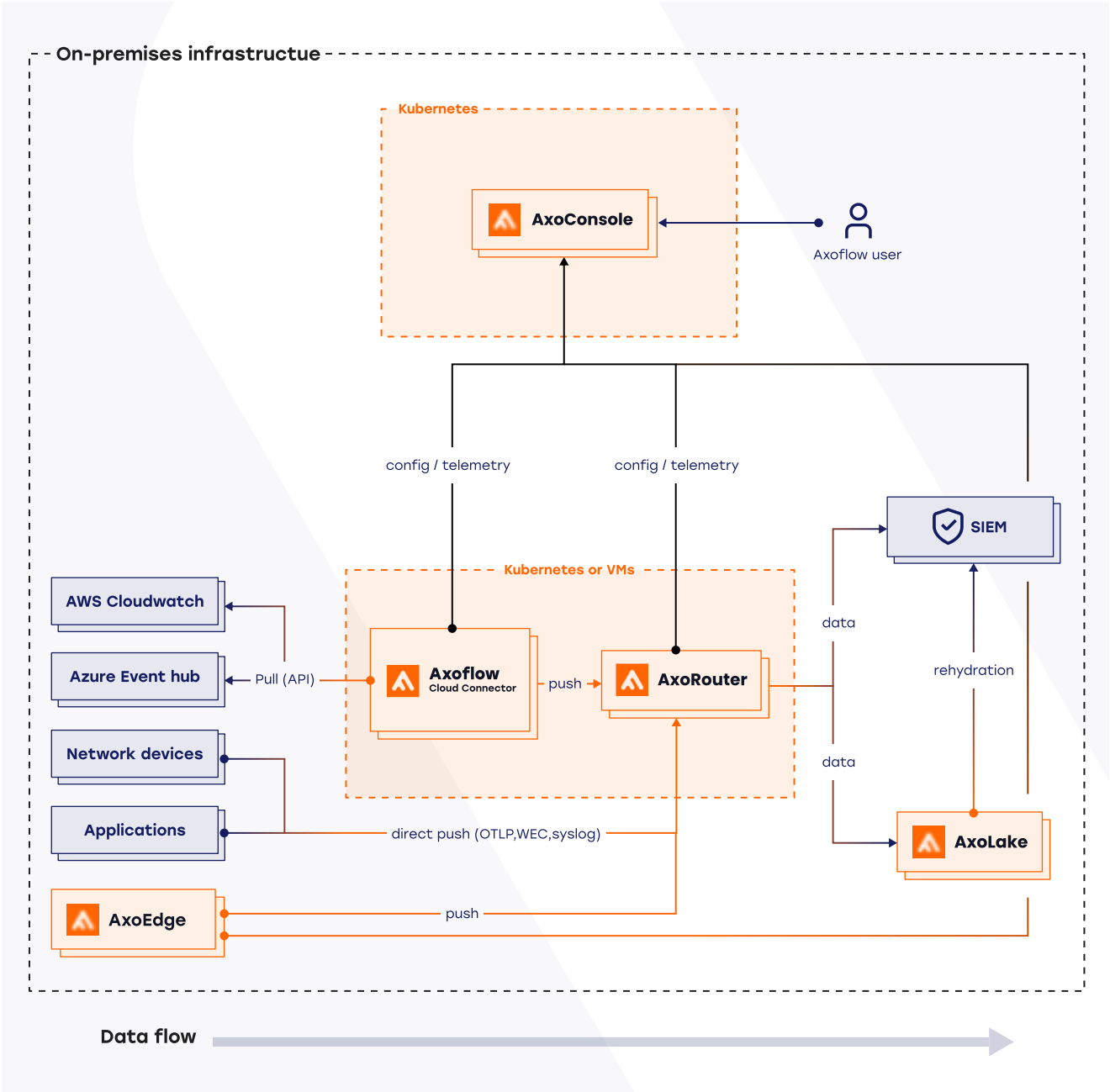
- Reliable transport: Axoflow uses the OpenTelemetry protocol (OTLP) between components (Axoflow agent to AxoRouter and Axoflow Cloud Connector to AxoRouter), so no messages are lost when crossing network boundaries between environments.
- Unified metrics: Every agent and router emits detailed per-component metrics, so you have end-to-end visibility into your data pipeline across all environments.
- Flexible routing: You can route data to environment-specific destinations (for example, a regional SIEM or local archive) or aggregate it centrally – or both – using policy-based routing.

4.7 On-prem or air-gapped deployment

Run AxoConsole, AxoRouter, and the Axoflow agents inside your own datacenter, without sending data or metrics outside.

Regulated industries – finance, healthcare, government – and security-sensitive organizations often cannot send data outside their own infrastructure and cannot install software from the internet. In these environments, routing logs and telemetry through a cloud service is not an option, and internet connectivity may be restricted or entirely unavailable.

Axoflow supports fully on-premises deployments, when the AxoConsole, AxoRouter nodes, and Axoflow agents all run inside your own datacenter. No data reaches external infrastructure, and the AxoConsole that manages the entire pipeline is also hosted on your own hardware. Air-gapped environments with no internet access are supported.



This approach gives you:

- Data sovereignty: all security data stays within your infrastructure. No component in the pipeline requires connectivity to Axoflow's cloud services.
- Air-gapped support: you can deploy and operate Axoflow in environments with no internet access.
- Single management plane: the on-premises AxoConsole configures and monitors all Axoflow agents and AxoRouter nodes, giving you the same visibility and control as the SaaS offering.
- Local aggregation: AxoRouters filter, enrich, and reduce data before it moves between segments of your network, minimizing internal traffic.

- Reliable transport: Axoflow uses the OpenTelemetry protocol (OTLP) between Axoflow agent and AxoRouter, so no messages are lost even under network instability.
- Unified metrics: every agent and router emits detailed per-component metrics, giving you end-to-end visibility into your data pipeline without leaving your perimeter.

For deployment instructions, see On-premise.

5 Concepts

The main concepts of Axoflow: automatic data processing, classification and reduction, normalization, policy-based routing, the host inventory, and reliable transport.

These pages explain how Axoflow works, and why. To see the path that data takes through the pipeline, start with Processing elements. The other pages explain the features that act on the data along that path:

- Automatic data processing and Classify and reduce security data: how Axoflow identifies, repairs, and reduces your data.
- Normalization: how Axoflow converts the data to the schema of the destination.
- Host attribution and inventory and Policy based routing: how labels about your hosts let you route data by what the source is.
- Reliable transport: how the components of Axoflow send data to each other.

For the components themselves, see Axoflow architecture.

5.1 Processing elements

The stages data passes through in Axoflow: sources, edge rules, AxoRouter connectors and classification, flows, and destinations.

Axoflow processes the data transported in your security data pipeline in the following stages:

1. Sources: Data enters the pipeline from a data source. A data source can be an external appliance or application, or a log collector agent managed by Axoflow.
   - Sources are hosts that are sending data to a data aggregator, like AxoRouter.
   - Edges are source hosts that are running a collector agent managed by AxoConsole, or have an Axolet agent reporting metrics from the host.

   For edge hosts, you can create:
   - collection rules that collect local data (for example, from log files, or Windows Event Log channels), and
   - data forwarding rules to transport the collected data to a router.
2. Custom metadata on the source: You can configure Axoflow to automatically add custom metadata to the data received from a source.
3. Router: The AxoRouter data aggregator processes the data it receives from the sources:
   - Connector: AxoRouter hosts receive data using source connectors. The different connectors are responsible for different protocols (like Syslog or OpenTelemetry). Some metadata labels are added to the data based on the connector it was received.
   - Metadata: AxoRouter classifies and identifies the incoming messages and adds metadata, for example, the vendor and product of the identified source.
   - Data extraction: AxoRouter extracts the relevant information from the content of the messages, and makes it available as structured data.

   The router can perform other processing steps, as configured in the flows that apply to the specific router (see next step).
4. Flow: You can configure flows in the AxoConsole that Axoflow uses to configure the AxoRouter instances to filter, route, and process the security data. Flows also allow you to automatically remove unneeded or redundant information from the messages, reducing data volume and SIEM and storage costs.
5. Destination: The router sends data to the specified destination in a format optimized for the specific destination.

5.2 Automatic data processing

Why Axoflow fixes malformed logs, missing timestamps, and invalid hostnames in the pipeline instead of in the SIEM.

When forwarding data to your SIEM, poor quality data and malformed logs that lack critical fields like timestamps or hostnames need to be fixed. The usual solutions fix the problem in the SIEM, and involve complex regular expressions, which are difficult to create and maintain. SIEM users often rely on vendors to manage these rules, but support is limited, especially for less popular devices. Axoflow offers a unique solution by automatically processing, curating, and classifying data before it's sent to the SIEM, ensuring accurate, structured, and optimized data, reducing ingestion costs and improving SIEM performance.

The problem

The main issue related to classification is that many devices send malformed messages: missing timestamp, missing hostname, invalid message format, and so on. Such errors can cause different kinds of problems:

- Log messages are often routed to different destinations based on the sender hostname. Missing or invalid hostnames mean that the message is not attributed to the right host, and often doesn't arrive at its intended destination.
- Incorrect timestamp or timezone hampers investigations during an incident, resulting in potentially critical data failing to show up (or extraneous data appearing) in queries for a particular period.
- Invalid data can lead to memory leaks or resource overload in the processing software (and to unusable monitoring dashboards) when a sequence number or other rapidly varying field is mistakenly parsed as the hostname, program name, or other low cardinality field.

Overall, they decrease the quality of security data you're sending to your SIEM tools, which increases false positives, requires secondary data processing to clean, and increases query time – all of which ends up costing firms a lot more.

For instance, this is a log message from a `SonicWall` firewall appliance:

```
<133> id=firewall sn=C0EFE33057B0 time="2024-10-07 14:56:47 UTC" fw=172.18.88.37 pri=6 c=1024
m=537 msg="Connection Closed" f=2 n=316039228 src=192.0.0.159:61254:X1:
dst=10.0.0.7:53:X3:SIMILDC01 proto=udp/dns sent=59 rcvd=134 vpnpolicy="ELG Main"
```

A well-formed syslog message should look like this:

```
<priority>timestamp hostname application: message body
```

As you can see, the SonicWall format is completely invalid after the initial `<priority>` field. Instead of the timestamp, hostname, and application name comes the free-form part of the message (in this case a whitespace-separated key=value list). Unless you extract the hostname and timestamp from the content of this malformed message, you won't be able to reconstruct the course of events during a security incident.

Axoflow provides data processing, curation, and classification intelligence that's built into the data pipeline, so it processes and fixes the data before it's sent to the SIEM.

Our solution

Our data engine and database solution automatically processes the incoming data: AxoRouter recognizes and classifies the incoming data, applies device-specific fixes for the errors, then enriches and optimizes the formatting for the specific destination (SIEM). This approach has several benefits:

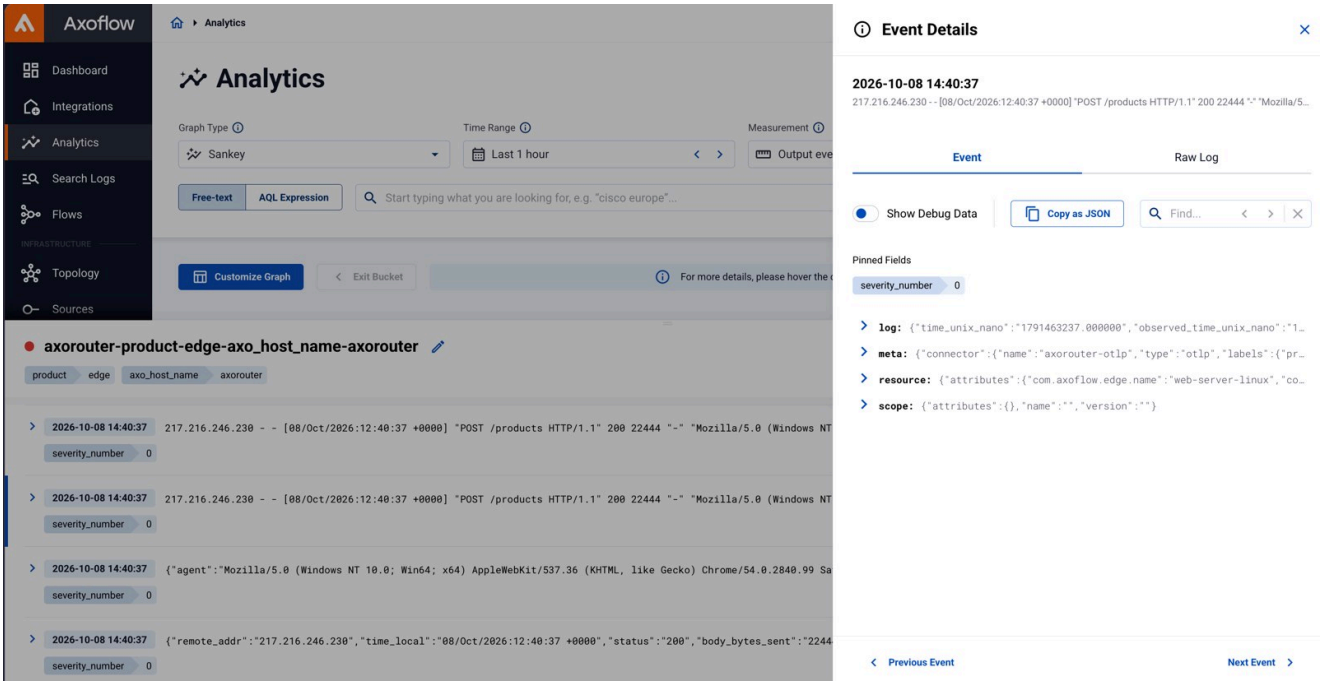
- Cost reduction: All data is processed before it's sent to the SIEM. That way, we can automatically reduce the amount of data sent to the SIEM (for example, by removing empty and redundant fields), cutting your data ingestion costs.
- Structured data: Axoflow recognizes the format of the incoming data payload (for example, JSON, CSV, LEEF, free text), and automatically parses the payload into a structured map. This allows us to have detailed, content-based metrics and alerts, and also makes it easy for you to add custom transformations if needed.
- SIEM-independent: The structured data representation allows us to support multiple SIEMs (and other destinations) and optimize the data for every destination.
- Performance: Compared to the commonly used regular expressions, it's more robust and has better performance, allowing you to process more data with fewer resources.
- Maintained by Axoflow: We maintain the database; you don't have work with it. This includes updates for new product versions and adding new devices. We proactively monitor and check the new releases of main security devices for logging-related changes and update our database. (If something's not working as expected, you can easily submit log samples and we'll fix it ASAP). Currently, we have over 150 application adapters in our database.

The automatic classification and curation also adds labels and metadata that can be used to make decisions and route your data. Messages with errors are also tagged with error-specific tags. For example, you can easily route all firewall and security logs to your Splunk deployment, and exclude logs (like debug logs) that have no security relevance and shouldn't be sent to the SIEM.

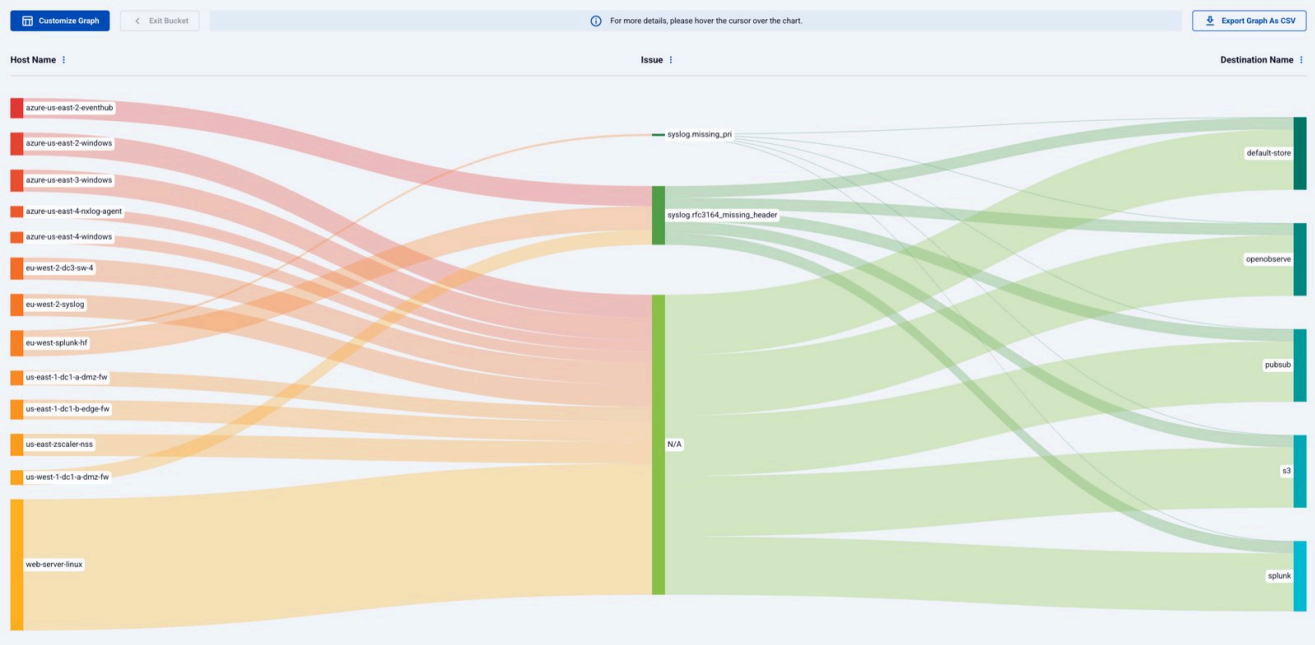
Message fixup

AxoConsole allows you to quickly drill down to find log flows with issues, and to tap into the log flow and see samples of the specific messages that are processed, along with the related parsing information, like tags that describe the errors of invalid messages.

- `message.utf8_sanitized` : The message is not valid UTF-8.
- `syslog.missing_timestamp` : The message has no timestamp.
- `syslog.invalid_hostname` : The hostname field doesn't seem to be valid, for example, it contains invalid characters.
- `syslog.missing_pri` : The priority (PRI) field is missing from the message.
- `syslog.unexpected_framing` : An octet count was found in front of the message, suggested invalid framing.
- `syslog.rfc3164_missing_header` : The date and the host are missing from the message – practically that's the entire header of RFC3164-formatted messages.
- `syslog.rfc5424_unquoted_sdata_value` : The message contains an incorrectly quoted RFC5424 SDATA field.
- `message.parse_error` : Some other parsing error occurred.



You can get an overview of such problems in your pipeline on the **Analytics** page: select **Customize Graph** and add the **Issue** label.


Customize Graph
Exit Bucket
For more details, please hover the cursor over the chart.
Export Graph As CSV
Host Name
Issue
Destination Name
azure-us-east-2-eventhub
azure-us-east-2-windows
azure-us-east-3-windows
azure-us-east-4-nxlog-agent
azure-us-east-4-windows
eu-west-2-dc3-sw-4
eu-west-2-syslog
eu-west-splunk-hf
us-east-1-dc1-a-dmz-fw
us-east-1-dc1-b-edge-fw
us-east-zscaler-nss
us-west-1-dc1-a-dmz-fw
web-server-linux
syslog.missing_pri
syslog.rfc3164_missing_header
N/A
default-store
openobserve
pubsub
s3
splunk

5.3 Host attribution and inventory

How labels in the Axoflow host inventory show the vendor, location, and owner of every source, and why that matters during an incident.

Axoflow's built-in inventory solution enriches your security data with critical metadata (like the origin host) so you can pinpoint the exact source of every data entry, enabling precise, label-based routing and more informed security decisions.

Enterprises and organizations collect security data (like syslog and other event data) from various data sources, including network devices, security devices, servers, and so on. When looking at a particular log entry during a security incident, it's not always trivial to determine what generated it. Was it an appliance or an application? Which team is the owner of the application or device? If it was a network device like a switch or a Wi-Fi access point (of which even medium-sized organizations have dozens), can you tell which one it was, and where is it located?

Cloud-native environments, like Kubernetes have addressed this issue using resource metadata. You can attach labels to every element of the infrastructure: containers, pods, nodes, and so on, to include region, role, owner or other custom metadata. When collecting log data from the applications running in containers, the log collector agent can retrieve these labels and attach them to the log data as metadata. This helps immensely to associate routing decisions and security conclusions with the source systems.

In non-cloud environments, like traditionally operated physical or virtual machine clusters, the logs of applications, appliances, and other data sources lack such labels. To identify the log source, you're stuck with the sender's IP address, the sender's hostname, and in some cases the path and filename of the log file, plus whatever information is available in the log message.

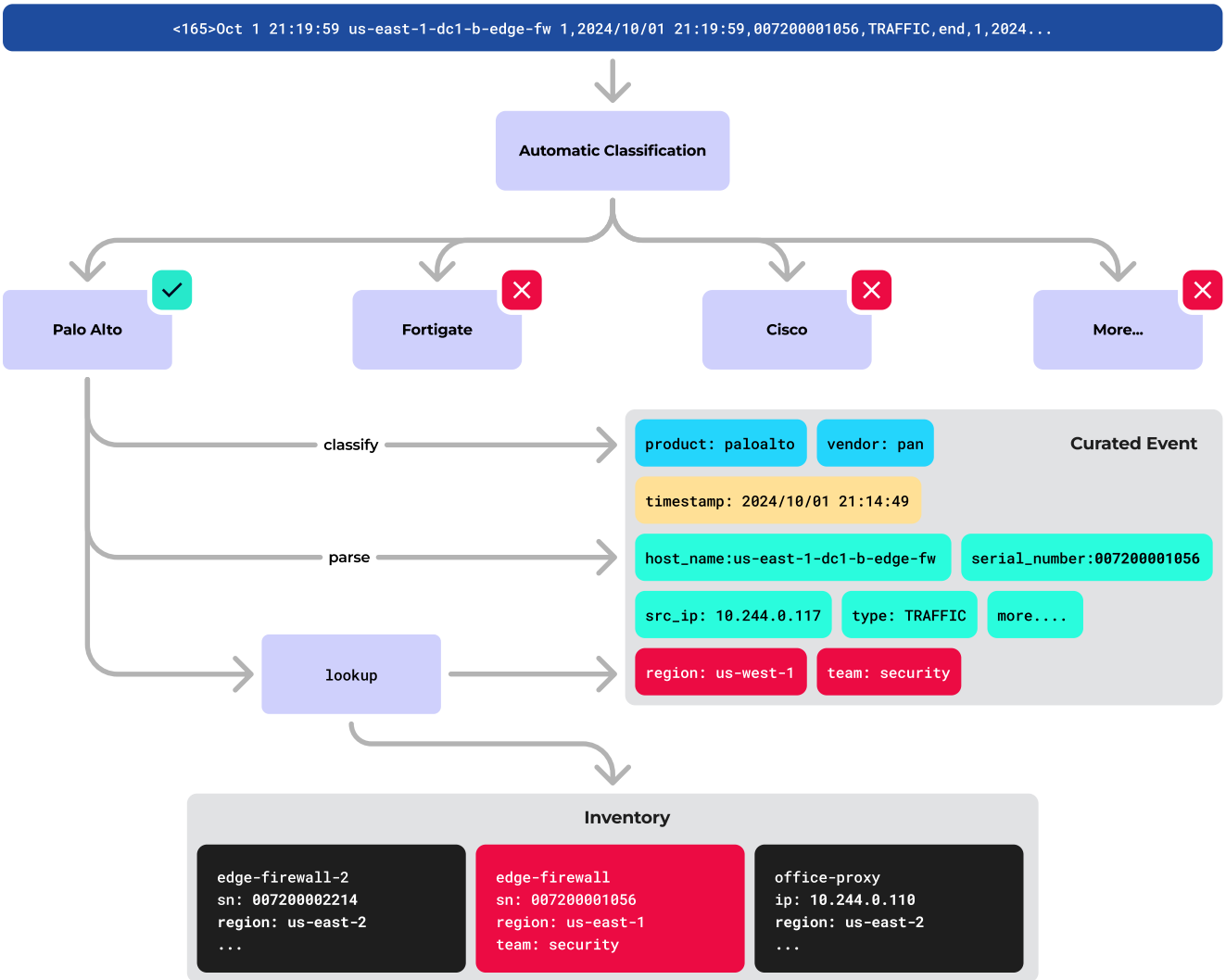
Why isn't the IP address or hostname enough?

- Source IP addresses are poor identifiers, as they aren't strongly tied to the host, especially when they're dynamically allocated using DHCP or when a group of senders are behind a NAT and have the same address.
- Some organizations encode metadata into the hostname or DNS record. However, compared to the volume of log messages, DNS resolving is slow. Also, the DNS record is available at the source, but might not be available at the log aggregation device or the log server. As a result, this data is often lost.

- Many data sources and devices omit important information from their messages. For example, the log messages of Cisco routers by default omit the hostname. (They can be configured to send their hostname, but unfortunately, often they aren't.) You can resolve their IP address to obtain the hostname of the device, but that leads back to the problem in the previous point.
- In addition, all the above issues are rooted in human configuration practices, which tend to be the main cause of anomalies in the system.

Correlating data to data source

Some SIEMs (like IBM QRadar) rely on the sender IP address to identify the data source. Others, like Splunk, delegate the task of providing metadata to the collector agent, but log sources often don't supply metadata, and even if they do, the information is based on IP address or DNS.



Certain devices, like SonicWall firewalls include a unique device ID in the content of their log messages. Having access to this ID makes attribution straightforward, but logging solutions rarely extract such information. AxoRouter does.

Axoflow builds an inventory to match the messages in your data flow to the available data sources, based on data including:

- IP address, host name, and DNS records of the source (if available),
- serial number, device ID, and other unique identifiers extracted from the messages, and
- metadata based on automatic classification of the log messages, like product and vendor name.

  Axoflow (or more precisely, AxoRouter) classifies the processed log data to identify common vendors and products, and automatically applies labels to attach this information.

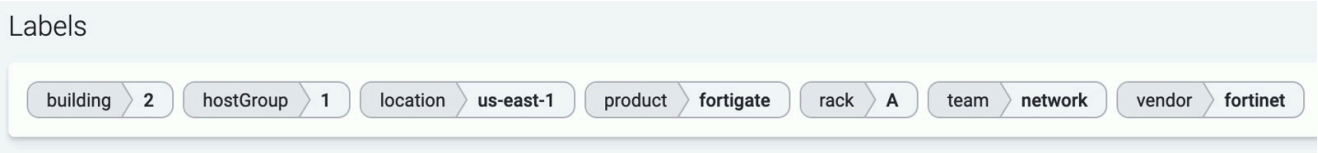
You can also integrate the telemetry pipeline with external asset management systems to further enrich your security data with custom labels and additional contextual information about your data sources.

5.4 Policy based routing

Route data by the vendor, product, location, or owner of its source, instead of by IP address or application name.

Axoflow's host attribution gives you unprecedented possibilities to route your data. You can tell exactly what kind of data you want to send where. Not only based on technical parameters like sender IP or application name, but using a much more fine-grained inventory that includes information about the devices, their roles, and their location. For example:

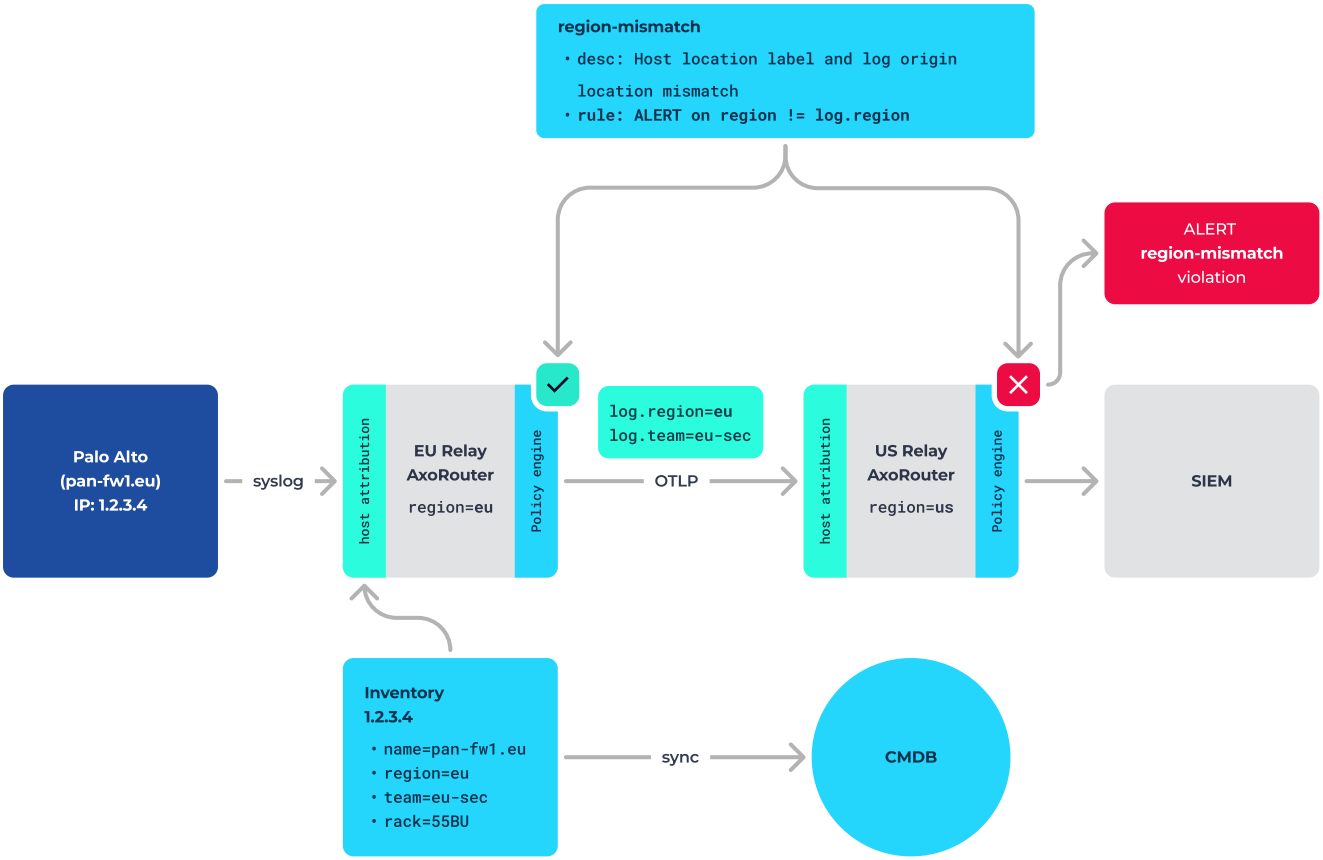
- The vendor and the product (for example, a SonicWall firewall)
- The physical location of the device (geographical location, rack number)
- Owner of the device (organization/department/team)



Axoflow automatically adds metadata labels based on information extracted from the data payload, but you can also add static custom labels to hosts manually, or dynamically using flows.

Paired with the information from the content of the messages, all this metadata allows you to formulate high-level, declarative policies and alerts using the metadata labels, for example:

- Route the logs of devices to the owner's Splunk index.
- Route every firewall and security log to a specific index.
- Let the different teams manage and observe their own devices.
- Granular access control based on relevant log content (for example, identify a proxy log-stream based on the upstream address), instead of just device access.
- Ensure that specific logs won't cross geographic borders, violating GDPR or other compliance requirements.


region-mismatch
• desc: Host location label and log origin location mismatch
• rule: ALERT on region != log.region
ALERT
region-mismatch
violation
Palo Alto
(pan-fw1.eu)
IP: 1.2.3.4
syslog
host attribution
EU Relay
AxoRouter
region=eu
Policy engine
log.region=eu
log.team=eu-sec
OTLP
host attribution
US Relay
AxoRouter
region=us
Policy engine
SIEM
Inventory
1.2.3.4
• name=pan-fw1.eu
• region=eu
• team=eu-sec
• rack=55BU
sync
CMDB

5.5 Classify and reduce security data

How Axoflow identifies each message by its content, and uses that classification to label the data and reduce its volume.

Axoflow provides a robust classification system that actually verifies the data it receives, instead of relying on using dedicated ports. This approach results in automatic data labeling, high-quality data, and volume reduction - out of the box, automated, without coding.

Classify the incoming data

Verifying which device or service a certain message belongs to is difficult: even a single data source (like an appliance) can have different kinds of messages, and you have to be able to recognize each one, uniquely. This requires:

- deep, device and vendor-specific understanding of the data, and also
- understanding of the syslog data formats and protocols, because oftentimes the data sources send invalid messages that you have to recognize and fix as part of the classification process.

Also, classification needs to be both reliable and performant. A naive implementation using regexps is neither, nevertheless, that's the solution you find at the core of today's ingestion pipelines. We at Axoflow understand that creating and maintaining such a classification database is difficult, this is why we decided to make classification a core functionality of the Axoflow Platform, so you will never need to write another parsing regexp. At the moment, Axoflow supports over 90 data sources of well-known vendors.

Classification and the ability to process your security data in the pipeline also allows you to:

- Fix the incoming data (like malformed firewall messages) to add missing information, like hostname or timestamp.
- Identify the source host,
- Parse the log to access the information contained within,
- Redact sensitive information before it gets sent to a SIEM or storage, like PII information,
- Reduce the data volume and as a result, storage and SIEM costs,
- Enrich the data with contextual information, like adding labels based on the source or content of the data,
- Use all the above to route the data to the appropriate destinations, and finally

- Transform the data into an optimized format that the destination can reliably and effortlessly consume. This includes mapping your data to multiple different schemas if you use multiple analytic tools.

Reduce data volume

Classifying and parsing the incoming data allows you to remove the parts that aren't needed, for example, to:

- drop entire messages if they are redundant or not relevant from a security perspective,
- remove parts of individual messages, like fields that are non-empty even if they do not convey information (for example, that contain values such as "N/A" or "0"), or
- remove redundant fields (where the same data appears multiple times in the message).

As this data reduction happens in the pipeline, before the data arrives in the SIEM or storage, it can save you significant costs, and also improves the quality of the data your detection engineers get to work with.

Palo Alto log reduction example

Let's see an example on how data reduction works in Axoflow. Here is a log message from a Palo Alto firewall:

```
<165>Mar 26 18:41:06 us-east-1-dc1-b-edge-fw 1,2025/03/26
18:41:06,007200001056,TRAFFIC,end,1,2025/03/26
18:41:06,192.168.41.30,192.168.41.255,10.193.16.193,192.168.41.255,allow-
all,,,netbios-ns,vsys1,Trust,Untrust,ethernet1/1,ethernet1/2,To-Panorama,2025/03/26
18:41:06,8720,1,137,137,11637,137,0x400000,udp,allow,276,276,0,3,2025/03/26
18:41:06,2,any,0,2345136,0x0,192.168.0.0-192.168.255.255,192.168.0.0-
192.168.255.255,0,3,0
```

Here's what you can drop from this particular message:

- **Redundant timestamps**: Palo Alto log messages contain up to five, practically identical timestamps (see the Receive time, Generated time, and High resolution timestamp fields in the Traffic Log Fields documentation):
  - the syslog timestamp in the header (Mar 26 18:41:06),

    - the time Panorama (the management plane of Palo Alto firewalls) collected the message (2025/03/26 18:41:06), and
    - the time when the event was generated (2025/03/26 18:41:06).

  The sample log message has five timestamps. Leaving only one timestamp can reduce the message size by up to 15%.

- The priority field (`<165>`) is identical in most messages and has no information value. While that takes up only about 1% of the size of the, on high-traffic firewalls even this small change adds up to significant data saving.

- Several fields contain default or empty values that provide no information, for example, default internal IP ranges like `192.168.0.0-192.168.255.255` . Removing such fields yields over 10% size reduction.

Note that when removing fields, we can delete only the value of the field, because the message format (CSV) relies on having a fixed order of columns for each message type. This also means that we have to individually check what can be removed from each of the 17 Palo Alto log type.

Palo Alto firewalls send this specific message when a connection is closed. They also send a message when a new connection is started, but that doesn't contain any information that's not available in the ending message, so it's completely redundant and can be dropped. As every connection has a beginning and an end, this alone almost halves the size of the data stored per connection. For example:

Connection start message:

```
<113>Apr 11 10:58:18 us-east-1-dc1-b-edge-fw
1,10:58:18.421048,007200001056,TRAFFIC,end,1210,10:58:18.421048,192.168.41.30,192.1
68.41.255,10.193.16.193,192.168.41.255,allow-all,,,ssl,vsys1,trust-
users,untrust,ethernet1/2.30,ethernet1/1,To-Panorama,2020/10/09
17:43:54,36459,1,39681,443,32326,443,0x400053,tcp,allow,43135,24629,18506,189,2020/
10/09 16:53:27,3012,laptops,0,1353226782,0x8000000000000000,10.0.0.0-
10.255.255.255,United States,0,90,99,tcp-fin,16,0,0,0,,testhost,from-
policy,,,0,,0,,N/A,0,0,0,0,ace432fe-a9f2-5a1e-327a-91fdce0077da,0
```

Connection end message:

```
<113>Apr 11 10:58:18 us-east-1-dc1-b-edge-fw
1,10:58:18.421048,007200001056,TRAFFIC,end,1210,10:58:18.421048,192.168.41.30,192.1
68.41.255,10.193.16.193,192.168.41.255,allow-all,,,ssl,vsys1,trust-
users,untrust,ethernet1/2.30,ethernet1/1,To-Panorama,2020/10/09
17:43:54,36459,1,39681,443,32326,443,0x400053,tcp,allow,43135,24629,18506,189,2020/
10/09 16:53:27,3012,laptops,0,1353226782,0x8000000000000000,10.0.0.0-
10.255.255.255,United States,0,90,99,tcp-fin,16,0,0,0,,testhost,from-
policy,,,0,,0,,N/A,0,0,0,0,ace432fe-a9f2-5a1e-327a-91fdce0077da,0
```

You can enable data reduction in your data flows using the **Reduce** processing step, and see the amount of data received and transferred in the flow on the **Metrics** page of the flow.

5.6 Normalization

Why destination-friendly schemas like ECS and OCSF matter for detection, and how Axoflow normalizes data inside the pipeline.

Axoflow normalizes incoming data and delivers it in a destination-friendly format and schema - covering keys, data types, streams, or indexes - rather than ingesting raw blobs. This includes consistent timestamps and attributes that downstream search queries, alerts, and detection engineers can immediately use. When a destination supports a specific normalization framework, Axoflow applies it by default. For vendor-specific schemas, such as Dynatrace Grail's semantic dictionary, Axoflow adds native support to ensure seamless integration.

Why Normalization Matters

Logs and security data come in different formats, from different devices, even when referring to the same event, making it difficult to find, understand, and correlate the data. Security events and detection rules should be agnostic to how the data was collected or where it came from. Your detection engineers shouldn't care about what's the exact name of the field where your different firewalls store the IP of the client: all such events should refer to a standard name. However, it needs expertise and a lot of legwork to do that properly. There is no one-to-one mapping between sources and schemas. You need to check the schema descriptions, understand the original content and know how the data will be used for analysis. The normalized version of an event helps your security teams best in their detection and investigation, as it makes it easy to correlate events received from different vendors (or different devices).

Axoflow builds this intelligence in. By combining AI with our deep data and security expertise, Axoflow automatically normalizes your data in real time within the pipeline. It supports multiple normalization frameworks, including Elastic Common Schema (ECS) and Open Cybersecurity Schema Framework (OCSF), so your security teams can work with normalized data from the start, enabling them to write stronger detection rules and uncover insights faster. You can even send the same data to multiple destinations in the normalized format that the specific destination supports.

We create and maintain our normalization database using an AI-backed process supervised by our engineers, so development is rapid, while it maintains high-quality and consistency.

To sum up, normalizing data:

- simplifies data ingestion for your SIEM
- improves correlation and detection

- speeds up investigations
- decreases vendor lock-in and improves flexibility.

5.7 Reliable transport

Why Axoflow components use OpenTelemetry (OTLP) over gRPC between each other: compression, authentication, and acknowledgement.

Between its components, Axoflow transports data using the reliable OpenTelemetry protocol (OTLP) for high performance, and to avoid losing messages. Under the hood, OpenTelemetry uses gRPC, and has superior performance compared to other log transport protocols, consumes less bandwidth (because it uses protocol buffers), and also provides features like:

- on-the-wire compression
- authentication
- application layer acknowledgement
- batched data sending
- multi-worker scalability on the client and the server.

In addition, Axoflow components use disk buffers to store data on the local hard disk if the destination (for example, your SIEM) or the network connection to the destination becomes unavailable. When the connection is reestablished, it automatically sends the stored messages to the destination in the order they were received.

6 Provision pipeline elements

Deploy AxoRouter and AxoEdge instances, and register hosts running syslog-ng, AxoSyslog, SC4S, or other supported log collectors in AxoConsole.

The following sections describe how to register a logging host into the AxoConsole.

- For appliances that are specifically supported by Axoflow, see onboarding appliances.
- For devices that are running a log collector agent that Axoflow supports, see Hosts with supported collectors.
- To provision an AxoRouter instance, see AxoRouter.
- To provision an Axoflow agent instance on a host, see Windows host - agent based solution and Agent for Linux edge hosts.

Hosts with supported collectors

If the host is running one of the following log collector agents and you can install Axolet on the host to receive detailed metrics about the host, the agent, and data flow the agent processes.

1. Install Axolet on the host. For details, see Axolet.
2. Configure the log collector agent of the host to integrate with Axolet. For details, see the following pages:
    - AxoSyslog
    - Splunk Connect for Syslog (SC4S)
    - syslog-ng

6.1 AxoRouter

Install AxoRouter on Linux or Kubernetes, and set up a highly available pair of AxoRouter hosts with keepalived.

AxoRouter receives the data of your sources, classifies and curates it, and sends it to your destinations. Deploy at least one AxoRouter before you add sources and flows.

- To install AxoRouter on a Linux host or virtual machine, see Install AxoRouter on Linux.
- To install AxoRouter on a Kubernetes cluster, see Install AxoRouter on Kubernetes.
- To run two AxoRouter hosts that share a virtual IP address and fail over to each other, see Set up AxoRouter high availability with keepalived.

6.1.1 Install AxoRouter on Linux

Check the requirements, install AxoRouter on a Linux host with the one-liner from AxoConsole, and send your first logs to it.

AxoRouter is a key building block of Axoflow that collects, aggregates, transforms and routes all kinds of telemetry and security data automatically. AxoRouter for Linux includes a Podman container running AxoSyslog, Axolet, and other components.

To install AxoRouter on a Linux host, complete the following steps. For other platforms, see AxoRouter.

> **Note**
>
> **Note that AxoRouter and Axoflow agent collects detailed, real-time metrics about the data-flows – giving you observability over the health of the security data pipeline and its components. Your security data remains in your self-managed cloud or in your on-prem instance where your sources, destinations, Axoflow agents, and AxoRouters are running, only metrics are forwarded to AxoConsole.**

What the install script does

When you deploy AxoRouter, you run a command that installs the required software packages, configures them and sets up the connection with Axoflow.

The installer script installs the `axolet` packages, then executes the `configure-axolet` command with the right parameters. (If the packages are already installed, the installer will update them unless the `none` *package format* is selected when generating the provisioning command.)

The install script is designed to be run as root (sudo), but you can configure AxoRouter to run as a non-root user.

The installer script performs the following main steps:

- Executes prerequisite checks:
  - Tests the network connection with the console endpoints.
  - Checks if the operating system is supported.
    - Checks if `podman` is installed.
- Downloads and installs the `axorouter` RPM or DEB package.
  - The package contains the `axorouter-ctl` and `setup-axorouter` commands and the `axorouter.container` unit files for podman-systemd.

> **CAUTION:**
>
> The package uses docker and standard service units. On older systems (e.g. Ubuntu 22.04 or older) this results in the following limitations:
>
> - AxoWec, AxoStore services are not supported.
> - Some `axorouter-ctl` commands are not fully supported.
> - Proxy environment variables need to be added to the docker systemctl configuration manually.

- Executes the `setup-axorouter` command, which
  - Updates the environment variables used by the axorouter service.
  - Enables and starts the AxoRouter service.
- Downloads and installs the `axolet` RPM or DEB package.
  - The package contains the `axolet` and `configure-axolet` commands, and the `axolet.service` systemd unit file.
- The `configure-axolet` command is executed with a configuration snippet on its standard input which contains a token required for registering into the management platform. The command:
  - Writes the initial `/etc/axolet/config.json` configuration file.

    Note: if the file already exists it will only be overwritten if the *Overwrite config* option is enabled when generating the provisioning command.
  - Enables and starts the `axolet` service.

`axolet` performs the following main steps on its first execution:

- Generates and persists a unique identifier (GUID).
- Initiates a cryptographic handshake process to AxoConsole.
- AxoConsole issues a client certificate to AxoRouter, which will be stored in the above mentioned `config.json` file.
- The service waits for an approval on AxoConsole. Once you approve the host registration request, axolet starts to manage the local services and send telemetry data to AxoConsole. It keeps doing so as long as the agent is registered.

Prerequisites

- AxoRouter supports the following operating systems:
  - Ubuntu 24.04 and 26.04,
  - Red Hat Enterprise Linux 8.8 or newer, 9.4 or newer, and 10.

Other compatible and derivative distributions (for example, AlmaLinux or Rocky Linux) should work as well, but we don't test them explicitly.

- Podman must be installed on the host (`sudo yum install podman`)
- When using AxoRouter with an on-premises AxoConsole deployment, you must prepare the AxoRouter host

Resource requirements

For a deployment that handles up to 1TB/day log traffic (~14500 EPS) even with complex routing and processing configurations, we recommend:

- 4 vCPU
- 8 GB memory
- 45 GB disk / hour. AxoRouter buffers incoming log data on disk if the destination or the network connection to the destination becomes unavailable. With a 1TB/day throughput, you need at least 45 GB of disk buffer per hour to avoid putting backpressure on your sources. This doesn't include any disk for using AxoStore.

For more details on hardware sizing, contact our support team.

Resource requirements for AxoStore

If you want to enable AxoStore on the node, you'll need:

- `1TB` storage for the free tier, or
- the storage limit of your AxoStore subscription.

Memory and CPU requirements depend on the incoming data volume, the **Config profile** you want to use for your AxoStores, the expected rate of events, and the number and complexity of the queries you'll be running. The **Default** configuration profile is recommended only for small-scale deployments and evaluation use cases that receive about 1000 events per second.

- For occasional access to the data, when you're mostly using the storage as a data warehouse, we recommend 1:80 memory to storage ratio and 6:1 GiB of memory to number of CPU cores ratio. That is, 12GiB memory and 2 CPU cores per 1TB of storage. At least 8GiB of RAM is recommended.
- If you're running frequent analytics and several concurrent queries on the stored data, we recommend 1:60 memory to storage ratio and 4:1 GiB of memory to CPU core ratio. That is, 16GiB memory and 4 CPU cores per 1TB of storage.

For detailed sizing recommendations, contact our support team.

The storage must be available on the volume storing the `/var/lib/clickhouse` folder. Alternatively, you can set a different directory to store your AxoStores using the AxoStore data directory option.

Network access

The host must be able to access the following domains related to the AxoConsole:

- When using AxoConsole SaaS:
  - `<your-tenant-id>.cloud.axoflow.io` : HTTPS traffic on TCP port 443, needed to download the binaries for Axoflow software (like Axolet and AxoRouter).
  - `kcp.<your-tenant-id>.cloud.axoflow.io` : HTTPS (mutual TLS) traffic on TCP port 443 for management traffic.
  - `telemetry.<your-tenant-id>.cloud.axoflow.io` : HTTPS (mutual TLS) traffic on TCP port 443, where Axolet sends the metrics of the host.
  - `us-docker.pkg.dev` : HTTPS traffic on TCP port 443, but only if you're forcing the installation script to pull the container images from the public repository. The Axolet and AxoRouter for Linux installation scripts download the images directly from AxoConsole.
- When using an on-premise AxoConsole:
  - The following domains should point to AxoConsole IP address to access Axoflow from your desktop and AxoRouter hosts:
    - `your-host.your-domain` : The main domain of your AxoConsole deployment.
    - `idp.your-host.your-domain` : A subdomain for the identity provider.
    - `kcp.your-host.your-domain` : A subdomain for management traffic.
    - `telemetry.your-host.your-domain` : A subdomain for agent telemetry.
  - The AxoConsole host must have the following **Open Ports**:
    - 80 TCP: Traefik ingress (HTTP, redirected to HTTPS).
    - 443 TCP: HTTPS for the AxoConsole UI, API, and all web traffic.
- When installing Axoflow agent for Windows or Linux:
  - `github.com` : HTTPS traffic on TCP port 443, for downloading installer packages.

By default, AxoRouter accepts data on the following ports (unless you've modified the default connector rules):

- 514 UDP and TCP for RFC3164 (BSD-syslog) and RFC5424 (IETF-syslog) formatted traffic. AxoRouter automatically recognizes and handles both formats.

- 601 TCP (RFC 3195 - Reliable Delivery for syslog) for RFC5424 (IETF-syslog) and RFC3164 (BSD-syslog) formatted traffic. AxoRouter automatically recognizes and handles both formats.
- 6514 TCP for TLS-encrypted syslog traffic (RFC 5425).
- 4317 TCP for OpenTelemetry log data.
- 5085 TCP for the Windows Event Collector (WEC) service.
- 8088 TCP for the Splunk HEC receiver (optional, only if you forward Splunk HEC traffic to AxoRouter).
- 9200 TCP for the Elasticsearch Bulk API connector (optional, only if you forward data from Elastic agents to AxoRouter).

To receive data on other ports or other protocols, configure other connector rules for the AxoRouter host.

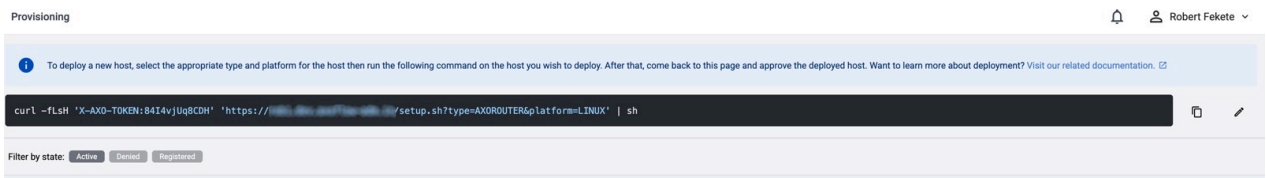
For TLS-encrypted syslog connections, create a new connector rule or edit an existing one, and configure the keys and certificates needed to encrypt the connections. For details, see Syslog.

> **Note**
>
> **When you create a connector rule on a new port, open that port in the firewall of of every corresponding AxoRouter as well.**

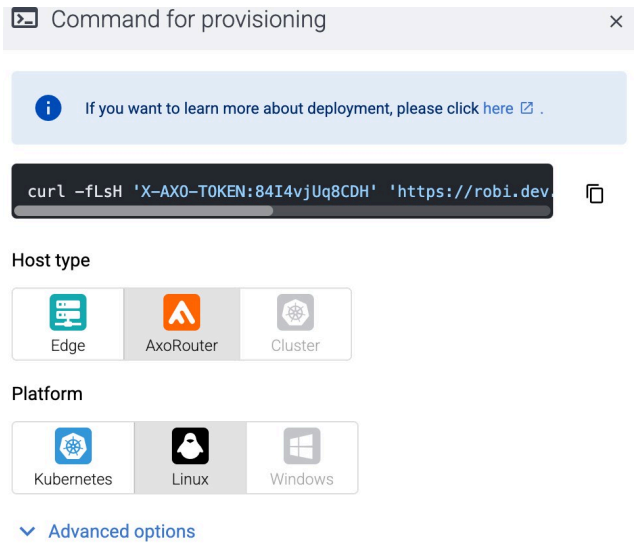
Install AxoRouter

> **Note**
>
> **When using AxoRouter with an on-premises AxoConsole deployment, you have to prepare the hosts before deploying AxoRouter. These steps are specific to on-premises AxoConsole deployments, and are not needed when using the SaaS AxoConsole.**

1. Select **Routers > Add Router**.



2. Select the platform (**Linux**). The one-liner installation command is displayed.



3. (Optional) If you don't want to store any logs locally on AxoRouter, disable AxoStore, select **Advanced options**, scroll down, and deselect **Enable AxoStore**.

4. (Optional)

   If needed, set the **Advanced options** (for example, proxy settings, or **External image pull** in air-gapped environments) to modify the installation parameters. Usually, you don't have to use advanced options unless the Axoflow support team instructs you to do so.

5. Open a terminal on the host where you want to install AxoRouter.

6. Run the one-liner, then follow the on-screen instructions.

> **Note**
>
> **Running the provisioning command with `sudo` would mask environment variables of the calling shell. Either start the whole procedure from a root shell, or let the install script call sudo when it needs to. In other words: don't add the `sudo` command to the provisioning command.**

Example output:

```
Do you want to install AxoRouter now? [Y]

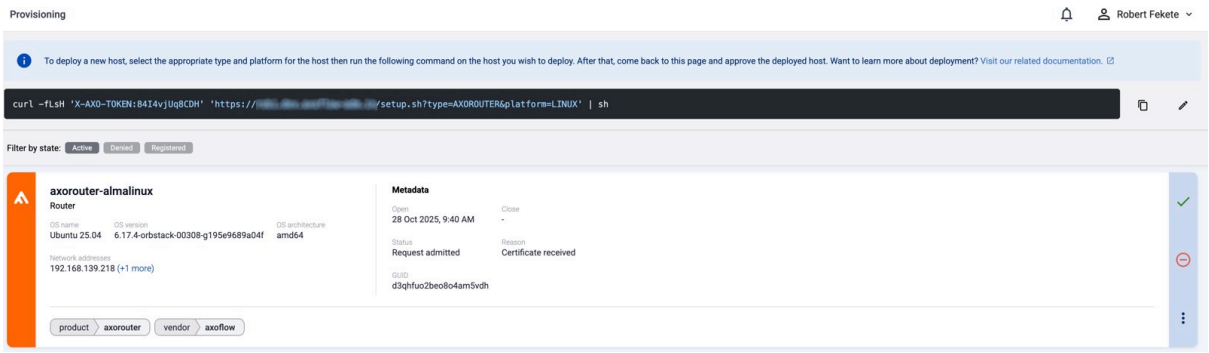
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  5480  100  5480    0     0  32076      0 --:--:-- --:--:-- --:--:-- 33414
Selecting previously unselected package axorouter.
(Reading database ... 17697 files and directories currently installed.)
Preparing to unpack axorouter.deb ...
Unpacking axorouter (0.66.0) ...
Setting up axorouter (0.66.0) ...
Low maximum socket receive buffer size value detected: 7500000 bytes (7.2MB).
Do you you want to permanently set the net.core.rmem_max sysctl value to
33554432 bytes (32MB) on this system? [Y]

net.core.rmem_max = 33554432
Created symlink '/etc/systemd/system/multi-user.target.wants/axostore.path' →
'/etc/systemd/system/axostore.path'.
Created symlink '/etc/systemd/system/multi-user.target.wants/axorouter-
wec.path' → '/etc/systemd/system/axorouter-wec.path'.
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 42.9M  100 42.9M    0     0  28.1M      0  0:00:01  0:00:01 --:--:-- 28.2M
Selecting previously unselected package axolet.
(Reading database ... 17707 files and directories currently installed.)
Preparing to unpack axolet.deb ...
Unpacking axolet (0.66.0) ...
Setting up axolet (0.66.0) ...
Created symlink '/etc/systemd/system/multi-user.target.wants/axolet.service' →
'/usr/lib/systemd/system/axolet.service'.
Now continue with onboarding the host on the Axoflow web UI.
```

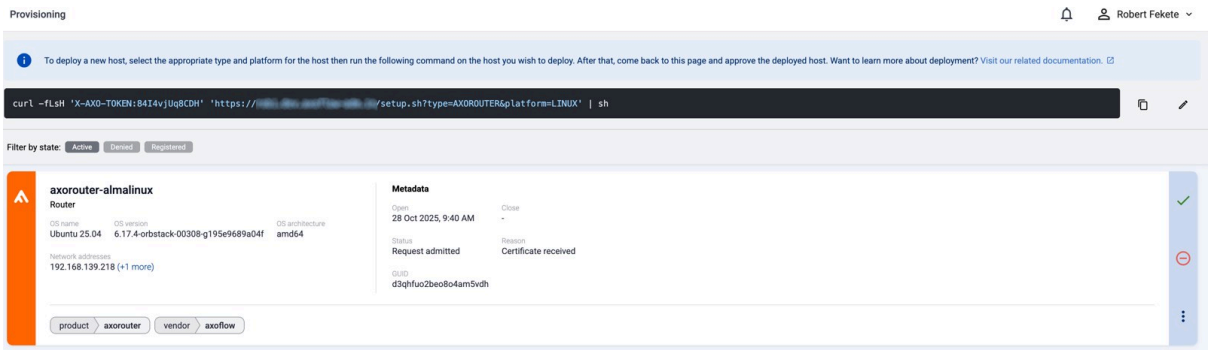
> **Note**
>
> If you get the following errors:
>
> ```
> AXOROUTER_FUSE envvar is not set. It may mean that the
> /usr/local/bin/setup-axorouter script is called without proper
> configuration.
> Failed to configure AxoRouter.
> Failed to deploy the AxoRouter.
> ```
>
> Set the following (or equivalent) in the `sudoers` file on the host, and rerun the installation command.
>
> ```
> Defaults!/usr/local/bin/setup-axorouter env_keep += "AXO* http_proxy
> https_proxy no_proxy"
> Defaults!/usr/local/bin/axorouter-ctl env_keep += "AXO* http_proxy
> https_proxy no_proxy"
> Defaults!/usr/local/bin/configure-axolet env_keep += "AXO* http_proxy
> https_proxy no_proxy"
> ```

7. Register the host.

    1. Reload the **Provisioning** page. There should be a registration request for the new AxoRouter deployment. Select ✓.

    2. Select **Register** to register the host. You can add a description and labels (in `label:value` format) to the host.



3. If the primary IP address (the first IP address shown in the **Network addresses** section on the **Routers** page for each AxoRouter) is not accessible from your edge hosts, set a **Network address override** (IP address or an FQDN) that's accessible. Otherwise, data forwarding from edge hosts will fail.

4. Select the **Topology** page. The new AxoRouter instance is displayed.

Run AxoRouter as non-root

By default, the install script runs AxoRouter as root. To configure it to run under a non-root user instead, see Run AxoRouter as non-root.

Create a flow

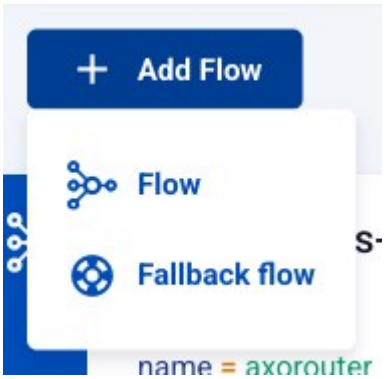
1. If you haven't already done so, create a new destination. If you've enabled AxoStore on the node and want to send data into AxoStore, see AxoStore.

2. **Create a flow to connect the new AxoRouter to the destination.**

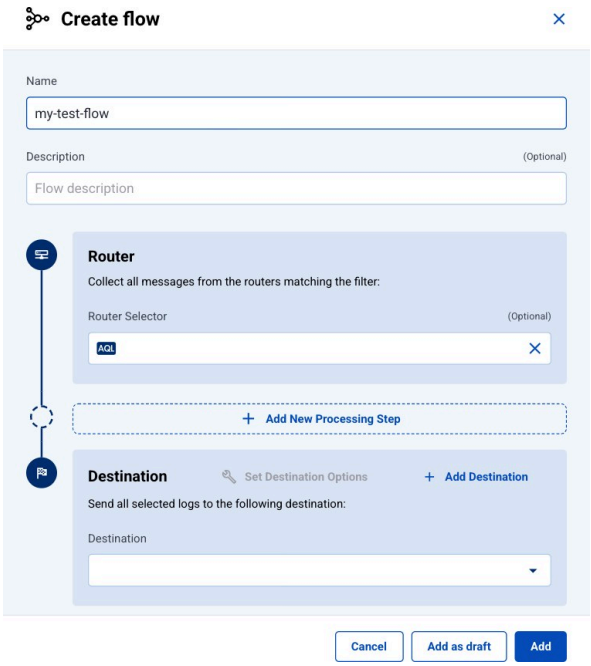
   1. Select **Flows**.

   2. Select **Add Flow > Flow**.

      To create a fallback flow, select **Add Flow > Fallback flow**.

   3. Enter a name for the flow, for example, `my-test-flow`.



4. In the **Router Selector** field, enter an expression that matches the routers you want to apply the flow to. To select a specific router, use a name selector, for example, `name = my-axorouter-hostname`.

   You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

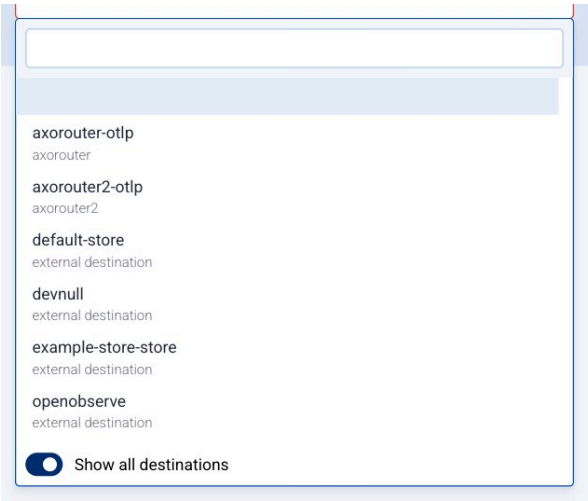
   - If you leave the router selector empty, it matches every AxoRouter instance.
   - To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.
   - If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

   > **Note**
   >
   > **You can configure multiple fallback flows, but only one fallback flow can apply to an AxoRouter (so the Router Selector of the fallback flows can't overlap).**

5. Select the **Destination** where you want to send your data. If you don't have any destination configured, you can select **Add Destination** in the destination section to create a new destination now. For details on the different destinations, see Destinations.

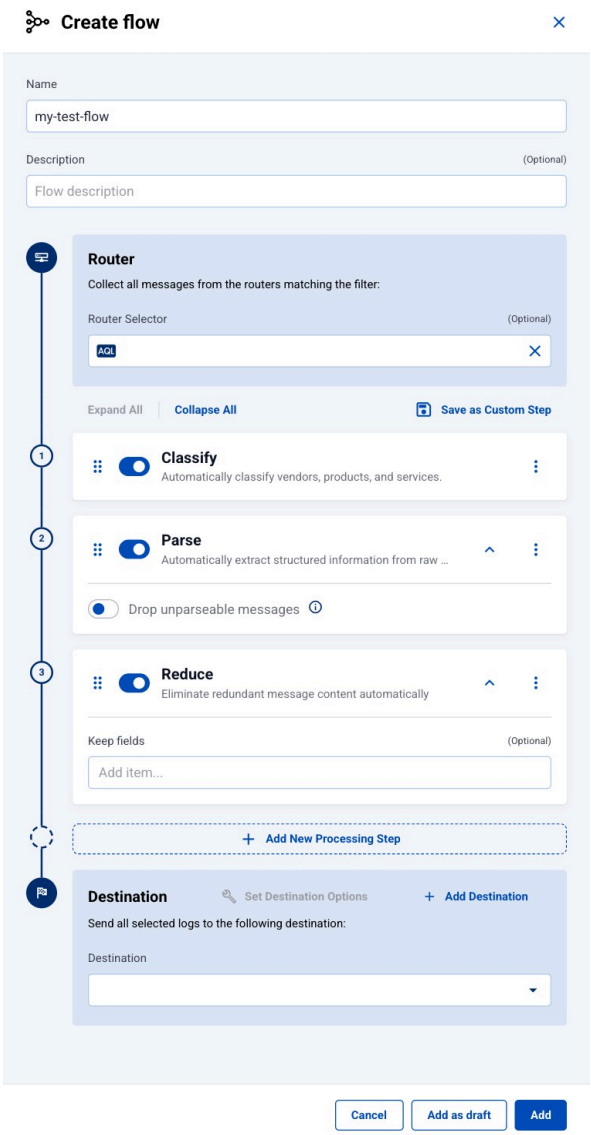
   - If you don't have any destination configured, see Destinations.

- If you've already created a store, it's automatically available as a destination. Note that the **Router Selector** of the flow must match only AxoRouters that have the selected store available, otherwise you'll get an error message.
- If you want to send data to another AxoRouter, enable the **Show all destinations** option, and select the connector of the AxoRouter where you want to send the data.



6. (Optional) To process the data transferred in the flow, select **Add New Processing Step**. For details, see Processing steps. For example:

   1. Add a **Classify**, a **Parse**, and a **Reduce** step, in that order, to automatically remove redundant and empty fields from your data.
   2. To select which messages are processed by the flow, add a **Select Messages** step, and enter a filter into the **AQL Expression** field. For example, to select only the messages received from Fortinet FortiGate firewalls, use the `meta.vendor = fortinet AND meta.product = fortigate` query.



7. Select **Add**, then confirm with **Add**. To save the flow without provisioning it to the routers yet, select **Add as draft** instead.

8. The new flow appears in the **Flows** list.



Send logs to AxoRouter

Configure your hosts to send data to AxoRouter.

- For appliances that are specifically supported by Axoflow, see Sources.
- For other appliances and generic Linux devices, see Generic tips.

- For a quick test without an actual source, you can also do the following (requires `nc` to be installed on the AxoRouter host):

  1. Open the AxoConsole, select **Topology**, then select the AxoRouter instance you've deployed. Alternatively, select `⌘/Ctrl + K` and enter the name of the AxoRouter.
  2. Select ⋮ **Host Options > Tap log flow > Input log flow**. Select **Start**.
  3. Open a terminal on your AxoRouter host.
  4. Run the following command to send 120 test messages (2 per second) in a loop to AxoRouter:

     ```
     for i in `seq 1 120`; do echo "<165> fortigate date=$(date -u +%Y-%m-%d)
     time=$(date -u +"%H:%M:%S%Z") devname=us-east-1-dc1-a-dmz-fw
     devid=FGT60D4614044725 logid=0100040704 type=event subtype=system
     level=notice vd=root logdesc=\"System performance statistics\"
     action=\"perf-stats\" cpu=2 mem=35 totalsession=61 disk=2 bandwidth=158/138
     setuprate=2 disklograte=0 fazlograte=0 msg=\"Performance statistics:
     average CPU: 2, memory:  35, concurrent sessions:  61, setup-rate: 2\"";
     sleep 0.5; done | nc -v 127.0.0.1 514
     ```

     Alternatively, you can send logs in an endless loop:

     ```
     while true; do echo "<165> fortigate date=$(date -u +%Y-%m-%d) time=$(date
     -u +"%H:%M:%S%Z") devname=us-east-1-dc1-a-dmz-fw devid=FGT60D4614044725
     logid=0100040704 type=event subtype=system level=notice vd=root
     logdesc=\"System performance statistics\" action=\"perf-stats\" cpu=2
     mem=35 totalsession=61 disk=2 bandwidth=158/138 setuprate=2 disklograte=0
     fazlograte=0 msg=\"Performance statistics: average CPU: 2, memory:  35,
     concurrent sessions:  61, setup-rate: 2\""; sleep 1; done | nc -v 127.0.0.1
     514
     ```

Manage AxoRouter

This section describes how to start, stop and check the status of the AxoRouter service on Linux.

Start AxoRouter

To start AxoRouter, execute the following command. For example:

```
systemctl start axorouter
```

If the service starts successfully, no output will be displayed.

The following message indicates that AxoRouter cannot start (see Check AxoRouter status):

```
Job for axorouter.service failed because the control process exited with error
code. See `systemctl status axorouter.service` and `journalctl -xe` for details.
```

Stop AxoRouter

To stop AxoRouter

1. Execute the following command.

   ```
   systemctl stop axorouter
   ```

2. Check the status of the AxoRouter service (see Check AxoRouter status).

Restart AxoRouter

To restart AxoRouter, execute the following command.

```
systemctl restart axorouter
```

Reload the configuration without restarting AxoRouter

To reload the configuration file without restarting AxoRouter, execute the following command.

```
systemctl reload axorouter
```

Check the status of AxoRouter service

To check the status of AxoRouter service

1. Execute the following command.

```
systemctl --no-pager status axorouter
```

2. Check the `Active:` field, which shows the status of the AxoRouter service. The following statuses are possible:
   - `active (running)` - `axorouter` service is up and running
   - `inactive (dead)` - `axorouter` service is stopped

Upgrade AxoRouter

AxoConsole raises an alert for the host when a new AxoRouter version is available. To upgrade to the new version, re-run the one-liner installation command you used to install AxoRouter, or select **Provisioning > Select type and platform** to create a new one.

Note that it can take a few minutes until the version numbers of the different services get updated on AxoConsole.

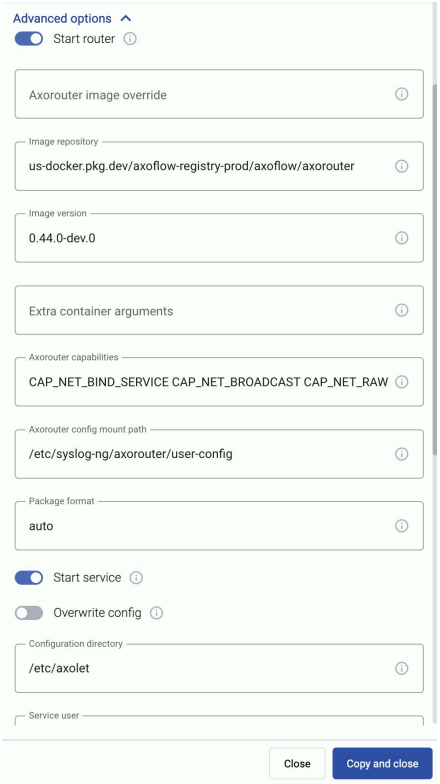
Uninstall AxoRouter

To remove AxoRouter from the host, see Uninstall AxoRouter from Linux.

6.1.1.1 Advanced installation options

Proxy settings, container and image options, and Axolet parameters for installing AxoRouter on Linux.

When installing AxoRouter, you can set a number of advanced options if needed for your environment. Setting the advanced options in the AxoConsole automatically updates the one-liner command that you can copy and run.



Alternatively, before running the one-liner you can use one of the following methods:

- Set the related environment variable for the option. For example:

```
export AXO_USER=syslogng
export AXO_GROUP=syslogng
```

- Set the related URL parameter for the option. For example:

```
curl -fLsH 'X-AXO-TOKEN:random-generated' 'https://<your-tenant-
id>.cloud.axoflow.io/setup.sh?
type=AXOROUTER&platform=LINUX&user=syslogng&group=syslogng' | sh
```

The AxoConsole doesn't show every option on the form. You can still set the hidden options using their environment variable or URL parameter:

- Some options are never shown on the form, for example, Docker arguments and Systemd override directory.
- The image override options are shown only if your tenant has a default value for them.
- The Logstore parameters are shown only if Logstore is enabled for your tenant.
- The RSigma parameters are shown only if detection is enabled for your tenant.

Proxy settings

Use the **HTTP proxy**, **HTTPS proxy**, and **No-proxy hosts** parameters to configure HTTP proxy settings. The installer, the Axolet service, and the AxoRouter services use these settings. To avoid using the proxy for the AxoRouter services, enable the **Skip proxy for the AxoRouter services** parameter as well. Lowercase variable names are preferred because they work universally.

Installation options

You can pass the following parameters to the installation script as environment variables, or as URL parameters.

> **Note**
>
> **Running the provisioning command with `sudo` would mask environment variables of the calling shell. Either start the whole procedure from a root shell, or let the install script call sudo when it needs to. In other words: don't add the `sudo` command to the provisioning command.**

External image pull

Default value:	`true`
Environment variable	`AXO_EXTERNAL_IMAGE_PULL`
URL parameter	`external_image_pull`

Description: If `true`, the installation script downloads the images from the central Axoflow repository. If `false`, the script downloads the images from the AxoConsole host. This is useful when deploying AxoRouters in airgapped environments.

AxoRouter capabilities

Default value:	`CAP_NET_BIND_SERVICE CAP_NET_BROADCAST CAP_NET_RAW CAP_SYSLOG CAP_BPF`
Environment variable	`AXO_AXOROUTER_CAPS`
URL parameter	`axorouter_caps`

Description: Capabilities added to the AxoRouter container.

AxoRouter config mount path

Default value:	`/etc/axorouter/user-config`
Environment variable	`AXO_CONFIG_MOUNT_INSIDE`
URL parameter	`config_mount_inside`

Description: Mount path for custom user configuration inside the AxoRouter container. The AxoRouter user config directory of the host is mounted to this path. This option isn't shown in the AxoConsole.

AxoRouter image override

Default value:	empty string
Environment variable	`AXO_IMAGE`
URL parameter	`image`

Description: Deploy the specified AxoRouter image.

AxoRouter user config directory

Default value:	`/etc/axorouter/user-config`
Environment variable	`AXO_CONFIG_MOUNT`

URL parameter	`config_mount`

Description: Directory on the host that stores the custom user configuration of AxoRouter. The directory is mounted to the AxoRouter config mount path inside the container.

Docker arguments for AxoRouter

Default value:	empty string
Environment variable	`AXO_DOCKER_ARGS`
URL parameter	`docker_args`

Description: Custom `docker` arguments for the AxoRouter container. Used only by the legacy (Docker-based) package. This option isn't shown in the AxoConsole. For the Podman-based package, use Podman arguments.

Image repository

Default value:	`us-docker.pkg.dev/axoflow-registry-prod/axoflow/axorouter`
Environment variable	`AXO_IMAGE_REPO`
URL parameter	`image_repo`

Description: Deploy AxoRouter from a custom image repository.

Image version

Default value:	Current Axoflow version
Environment variable	`AXO_IMAGE_VERSION`
URL parameter	`image_version`

Description: Deploy the specified AxoRouter version.

Legacy (Docker-based) package

Default value:	`auto`
Available values:	`auto` , `true` , `false`
Environment variable	`AXO_LEGACY_PACKAGE`
URL parameter	`legacy_package`

Description: If `true` , install the Docker-based `axorouter-legacy` package instead of the Podman-based package. If `auto` , the installer selects the legacy package on Ubuntu 22.04, and the Podman-based package on other platforms.

Non-interactive installation

Default value:	`false`
Available values:	`true` , `false`
Environment variable	`AXO_NON_INTERACTIVE`
URL parameter	`non_interactive`

Description: If `true` , the installation doesn't ask for confirmation. Use it for unattended installations.

Package format

Default value:	`auto`
Available values:	`auto` , `deb` , `rpm` , `tar` , `none`
Environment variable	`AXO_INSTALL_PACKAGE`
URL parameter	`install_package`

Description: File format of the installer package.

Persist directory

Default value:	`/var/lib/syslog-ng`
Environment variable	`AXO_PERSIST_MOUNT`

URL parameter	`persist_mount`

Description: Directory on the host that stores the persist file and the disk buffers of AxoRouter. The data directories of the WEC and the `axorouter-resolver` services are created under this directory. Make sure that there's sufficient disk space for the disk buffers.

If you store the AxoRouter data on a separate disk, set the AxoStore data directory and the Logstore directories as well, because they're independent of this directory.

Podman arguments for AxoRouter services

Default value:	empty string
Environment variable	`AXO_PODMAN_ARGS`
URL parameter	`podman_args`

Description: Custom `podman` arguments for the containers of the AxoRouter services: AxoRouter, AxoRouter WEC, AxoStore, `axorouter-resolver`, Logstore, RSigma, and Axoflow Cloud Connector. The Logstore container also gets the `--read-only` argument. You can specify only a single argument. If you leave this option empty, the installer uses `--log-level=warn`.

Raise socket receive buffer limit

Default value:	`auto`
Available values:	`auto`, `true`, `false`
Environment variable	`AXO_SET_RMEM_MAX`
URL parameter	`set_rmem_max`

Description: If `true`, the installer permanently raises the `net.core.rmem_max` sysctl value of the host if it's too low. If `auto`, the installer asks for confirmation when running in an interactive terminal, and raises the value without asking in a non-interactive installation. If `false`, the installer doesn't change the value.

Skip prerequisites check

Default value:	`false`
Available values:	`true` , `false`
Environment variable	`AXO_SKIP_PREFLIGHT_CHECK`
URL parameter	`skip_preflight_check`

Description: If `true` , the installer skips checking that the API server and the telemetry proxy are reachable, and that their certificates are valid.

Start router

Default value:	`true`
Available values:	`true` , `false`
Environment variable	`AXO_START_ROUTER`
URL parameter	`start_router`

Description: Start AxoRouter after installation.

Systemd override directory

Default value:	`/etc/systemd/system/axolet.service.d`
Environment variable	`AXO_SYSTEMD_OVERRIDE_DIR`
URL parameter	`systemd_override_dir`

Description: Directory for the systemd unit overrides of the Axolet service. This option isn't shown in the AxoConsole.

Axolet parameters

API server host

Default value:	
Environment variable	Not available (URL parameter only)

URL parameter	`api_server_host`

Description: Override the host part of the API endpoint for the host. Axolet connects to this host, but still uses the original hostname of the endpoint to verify the TLS certificate.

Skip proxy for the AxoRouter services

Default value:	`false`
Available values:	`true` , `false`
Environment variable	`AXO_AVOID_PROXY`
URL parameter	`avoid_proxy`

Description: If set to `true` , the AxoRouter services don't use the proxy. This option makes a difference only if you set a proxy. The installer and the Axolet service always use the proxy, because they reach the AxoConsole the same way as the installation command. (The legacy `y` value is the same as `true` .)

The option is disabled by default, because a destination can be outside your network (for example, Splunk Cloud), and the AxoRouter services need the proxy to reach it. Enable it if every destination is in your internal network. To use the proxy only for some destinations, leave this option disabled, and list your internal hosts in **No-proxy hosts** instead.

Axolet capabilities

Default value:	`CAP_SYS_PTRACE CAP_SYS_CHROOT`
Available values:	Whitespace-separated list of capability names with `CAP_` prefix.
Environment variable	`AXO_CAPS`
URL parameter	`caps`

Description: Ambient Linux capabilities the `axolet` service will use.

Configuration directory

Default value:	`/etc/axolet`
Environment variable	`AXO_CONFIG_DIR`
URL parameter	`config_dir`

Description: The directory where the configuration files are stored.

HTTP proxy

Default value:	empty string
Environment variable	`AXO_HTTP_PROXY`
URL parameter	`http_proxy`

Description: The HTTP proxy that the installer, the Axolet service, and the AxoRouter services use, unless you skip the proxy for the AxoRouter services.

HTTPS proxy

Default value:	empty string
Environment variable	`AXO_HTTPS_PROXY`
URL parameter	`https_proxy`

Description: The HTTPS proxy that the installer, the Axolet service, and the AxoRouter services use, unless you skip the proxy for the AxoRouter services.

Initial labels

Default value:	`platform=linux,product=axorouter,vendor=axoflow`
Environment variable	Not available (URL parameter only)
URL parameter	`initial_labels`

Description: Comma-separated list of `key=value` labels. These labels are set on the host registration request, and are suggested for the host when you add the source to AxoConsole. For example, `product=windows,team=windows,vendor=microsoft`

No-proxy hosts

Default value:	empty string
Environment variable	`AXO_NO_PROXY`
URL parameter	`no_proxy`

Description: Comma-separated list of hosts to reach directly instead of through the proxy. It applies to every component that uses the proxy. Loopback addresses are always excluded, regardless of this setting.

Overwrite config

Default value:	`false`
Available values:	`true` , `false`
Environment variable	`AXO_CONFIG_OVERWRITE`
URL parameter	`config_overwrite`

Description: If set to `true` , the configuration process will overwrite existing configuration (`/etc/axolet/config.json`). This means that the agent will get a new GUID and it will require approval on the AxoConsole.

Service group

Default value:	`root`
Environment variable	`AXO_GROUP`
URL parameter	`group`

Description: Name of the group and Axolet will be running as. It should be either `root` or the group `syslog-ng` is running as.

Service user

Default value:	`root`

Environment variable	`AXO_USER`
URL parameter	`user`

Description: Name of the user Axolet will be running as. It should be either `root` or the user `syslog-ng` is running as. See also Run axolet as non-root.

Start service

Default value:	`true`
Available values:	`true` , `false`
Environment variable	`AXO_START`
URL parameter	`start`

Description: Start axolet agent at the end of installation. Use false for preparing golden images. In this case `axolet` will generate a new GUID on the first boot after cloning the image.

WEC parameters

These parameters are related to the Windows Event Collector server that can be run on AxoRouter. For details, see Windows Event Collector (WEC).

WEC image override

Default value:	``
Environment variable	`AXO_WEC_IMAGE`
URL parameter	`wec_image`

Description: Deploy the specified WEC image.

WEC image repository

Default value:	`us-docker.pkg.dev/axoflow-registry-prod/axoflow/axorouter-wec`
Environment variable	`AXO_WEC_IMAGE_REPO`

URL parameter	`wec_image_repo`

Description: Deploy the Windows Event Collector server from a custom image repository.

WEC image version

Default value:	Current Axoflow version
Environment variable	`AXO_WEC_IMAGE_VERSION`
URL parameter	`wec_image_version`

Description: Deploy the specified Windows Event Collector server version.

`axorouter-resolver` parameters

These parameters are related to the resolver service that can be run on AxoRouter.

The `AXORESOLVER_*` environment variables (without the `AXO_` prefix) are read by the setup script of the AxoRouter package, not by the installer one-liner, so they don't have URL parameters and aren't available in the AxoConsole.

Enable axorouter-resolver

Default value:	`true`
Available values:	`true` , `false`
Environment variable	`AXO_AXORESOLVER_ENABLED`
URL parameter	`axoresolver_enabled`

Description: Install `axorouter-resolver` for this AxoRouter deployment.

axorouter-resolver image override

Default value:	
Environment variable	`AXO_AXORESOLVER_IMAGE`
URL parameter	`axoresolver_image`

Description: Deploy the specified `axorouter-resolver` image.

axorouter-resolver image repository

Default value:	`us-docker.pkg.dev/axoflow-registry-prod/axoflow/axoresolver`
Environment variable	`AXO_AXORESOLVER_IMAGE_REPO`
URL parameter	`axoresolver_image_repo`

Description: Deploy the `axorouter-resolver` from a custom image repository.

axorouter-resolver image version

Default value:	Current Axoflow version
Environment variable	`AXO_AXORESOLVER_IMAGE_VERSION`
URL parameter	`axoresolver_image_version`

Description: Deploy the specified `axorouter-resolver` version.

axorouter-resolver persistence directory

Default value:	`${AXOROUTER_PERSIST_MOUNT}/axoresolver`
Environment variable	`AXORESOLVER_PERSIST_DIR`
URL parameter	

Description: Directory where `axorouter-resolver` stores persistent data.

axorouter-resolver configuration directory

Default value:	`/etc/axorouter/axoresolver`
Environment variable	`AXORESOLVER_CONFIG_DIR`
URL parameter	

Description: Directory where `axorouter-resolver` configuration files are stored.

axorouter-resolver environment file

Default value:	`/etc/axorouter/axoresolver.env`
Environment variable	`AXORESOLVER_ENV_FILE`
URL parameter	

Description: Path to the `axorouter-resolver` environment file.

axorouter-resolver configuration file

Default value:	`${AXORESOLVER_CONFIG_DIR}/resolvers.json`
Environment variable	`AXORESOLVER_CONFIG_FILE`
URL parameter	

Description: Path to the `axorouter-resolver` configuration file.

axorouter-resolver listening socket

Default value:	`${AXORESOLVER_PERSIST_DIR}/axoresolver.sock`
Environment variable	`AXORESOLVER_LISTENING_SOCKET`
URL parameter	

Description: Path to the socket file that `axorouter-resolver` listens on.

axorouter-resolver control socket

Default value:	`${AXORESOLVER_PERSIST_DIR}/axoresolver-ctl.sock`
Environment variable	`AXORESOLVER_CONTROL_SOCKET_PATH`
URL parameter	

Description: Path to the socket file used to control `axorouter-resolver`.

axorouter-resolver cache file

Default value:	`${AXORESOLVER_PERSIST_DIR}/axoresolver-cache.json`
Environment variable	`AXORESOLVER_CACHE_FILE`
URL parameter	

Description: Path to the `axorouter-resolver` cache file.

axorouter-resolver cache write interval

Default value:	`10`
Environment variable	`AXORESOLVER_CACHE_WRITE_INTERVAL`
URL parameter	

Description: Interval for writing the cache to disk. If you don't specify the unit, it's treated as seconds: `10` is the same as `10s`. You can use other units as well, for example, `m` for minutes.

axorouter-resolver log level

Default value:	`INFO`
Environment variable	`AXORESOLVER_LOG_LEVEL`
URL parameter	

Description: Log level for `axorouter-resolver`.

AxoStore parameters

These parameters are related to the AxoStore that can be run on AxoRouter. For details, see Storage.

Enable AxoStore

Default value:	`true`
Available values:	`true` , `false`
Environment variable	`AXO_AXOSTORE_ENABLED`
URL parameter	`axostore_enabled`

Description: Install AxoStore for this AxoRouter deployment.

AxoStore image override

Default value:	empty string
Environment variable	`AXO_AXOSTORE_IMAGE`
URL parameter	`axostore_image`

Description: Deploy the specified AxoStore image instead of the default.

AxoStore image repository

Default value:	`us-docker.pkg.dev/axoflow-registry-prod/axoflow/axostore`
Environment variable	`AXO_AXOSTORE_IMAGE_REPO`
URL parameter	`axostore_image_repo`

Description: Deploy the AxoStore from a custom image repository.

AxoStore image version

Default value:	Current Axoflow version
Environment variable	`AXO_AXOSTORE_IMAGE_VERSION`
URL parameter	`axostore_image_version`

Description: Deploy the specified AxoStore version.

AxoStore data directory

Default value:	`/var/lib/clickhouse`
Environment variable	`AXO_AXOSTORE_DATA_DIR`
URL parameter	`axostore_data_dir`

Description: The directory on the host where the AxoStores are stored. Make sure that there's sufficient disk space, especially for long-time retention and/or high data volume. This directory is independent of the Persist directory.

If you don't set this option, the installer uses the value of the `AXOSTORE_MOUNT` environment variable if it's set, otherwise the default value.

Axoflow Cloud Connector parameters

These parameters are related to the Axoflow Cloud Connector services that can be run on AxoRouter.

Enable Cloudconnectors

Default value:	`true`
Available values:	`true` , `false`
Environment variable	`AXO_CLOUDCONNECTORS_ENABLED`
URL parameter	`cloudconnectors_enabled`

Description: Install Axoflow Cloud Connector for this AxoRouter deployment. Enabling this option is a prerequisite for sources that use Axoflow Cloud Connector, for example, Azure Event Hubs.

Cloudconnectors image override

Default value:	``
Environment variable	`AXO_CLOUDCONNECTORS_IMAGE`
URL parameter	`cloudconnectors_image`

Description: Deploy the specified Axoflow Cloud Connector image instead of the default.

Cloudconnectors image repository

Default value:	`us-docker.pkg.dev/axoflow-registry-prod/axoflow/axorouter-cloudconnectors`
Environment variable	`AXO_CLOUDCONNECTORS_IMAGE_REPO`
URL parameter	`cloudconnectors_image_repo`

Description: Deploy the Axoflow Cloud Connector from a custom image repository.

Cloudconnectors image version

Default value:	Current Axoflow version
Environment variable	`AXO_CLOUDCONNECTORS_IMAGE_VERSION`
URL parameter	`cloudconnectors_image_version`

Description: Deploy the specified Axoflow Cloud Connector version.

Logstore parameters

These parameters are related to the Logstore service that can be run on AxoRouter. Logstore is disabled by default. For details, see Create Logstore.

Logstore is an early-access feature. The AxoConsole shows these parameters only if Logstore is enabled for your tenant.

Enable Logstore

Default value:	`false`
Available values:	`true` , `false`
Environment variable	`AXO_LOGSTORE_ENABLED`
URL parameter	`logstore_enabled`

Description: Install Logstore for this AxoRouter deployment.

Logstore image override

Default value:	``
Environment variable	`AXO_LOGSTORE_IMAGE`
URL parameter	`logstore_image`

Description: Deploy the specified Logstore image instead of the default.

Logstore image repository

Default value:	`us-docker.pkg.dev/axoflow-registry-prod/axoflow/axorouter-logstore`
Environment variable	`AXO_LOGSTORE_IMAGE_REPO`
URL parameter	`logstore_image_repo`

Description: Deploy Logstore from a custom image repository.

Logstore image version

Default value:	Current Axoflow version
Environment variable	`AXO_LOGSTORE_IMAGE_VERSION`
URL parameter	`logstore_image_version`

Description: Deploy the specified Logstore version.

Logstore configuration directory

Default value:	`/etc/axorouter/logstore`
Environment variable	`AXO_LOGSTORE_CONFIG_HOST_DIR`
URL parameter	`logstore_config_host_dir`

Description: Directory on the host that stores the Logstore configuration that AxoConsole sends to AxoRouter.

Logstore data directory

Default value:	`/var/lib/logstore`
Environment variable	`AXO_LOGSTORE_DATA_HOST_DIR`
URL parameter	`logstore_data_host_dir`

Description: Directory on the host that stores the working data of Logstore: the write-ahead log, and the scheduler and statistics databases. This data must be on a single local filesystem, you can't split it to multiple disks.

Reinstalling AxoRouter doesn't migrate any existing data to the new default directory.

Logstore warm directory

Default value:	`<logstore-data-directory>/warm` (`/var/lib/logstore/warm`)
Environment variable	`AXO_LOGSTORE_WARM_HOST_DIR`
URL parameter	`logstore_warm_host_dir`

Description: Directory on the host that stores the merged, but not yet archived Logstore data (in Apache Parquet format). You can place this directory on a separate disk.

Logstore archive directory

Default value:	`<logstore-data-directory>/archive` (`/var/lib/logstore/archive`)
Environment variable	`AXO_LOGSTORE_ARCHIVE_HOST_DIR`
URL parameter	`logstore_archive_host_dir`

Description: Directory on the host that stores the archived Logstore data (in Apache Parquet format). You can place this directory on a separate, slower disk, or on a network volume.

Logstore directories inside the container

Default value:	`/var/lib/logstore` , `/var/lib/logstore/warm` , `/var/lib/logstore/archive`

Environment variable	`AXO_LOGSTORE_DATA_DIR` , `AXO_LOGSTORE_WARM_DIR` , `AXO_LOGSTORE_ARCHIVE_DIR`
URL parameter	`logstore_data_dir` , `logstore_warm_dir` , `logstore_archive_dir`

Description: Mount paths of the data, warm, and archive directories inside the Logstore container. These options aren't shown in the AxoConsole, and usually you don't need to change them. To move the Logstore data on the host, set the host directory options instead.

RSigma parameters

These parameters are related to the RSigma detection engine that can be run on AxoRouter. RSigma is disabled by default. The AxoConsole shows these parameters only if detection is enabled for your tenant.

Enable RSigma

Default value:	`false`
Available values:	`true` , `false`
Environment variable	`AXO_RSIGMA_ENABLED`
URL parameter	`rsigma_enabled`

Description: Install the RSigma detection engine for this AxoRouter deployment.

RSigma image override

Default value:	``
Environment variable	`AXO_RSIGMA_IMAGE`
URL parameter	`rsigma_image`

Description: Deploy the specified RSigma image instead of the default.

RSigma image repository

Default value:	`us-docker.pkg.dev/axoflow-registry-prod/axoflow/axorouter-rsigma`
Environment variable	`AXO_RSIGMA_IMAGE_REPO`
URL parameter	`rsigma_image_repo`

Description: Deploy RSigma from a custom image repository.

RSigma image version

Default value:	Current Axoflow version
Environment variable	`AXO_RSIGMA_IMAGE_VERSION`
URL parameter	`rsigma_image_version`

Description: Deploy the specified RSigma version.

6.1.1.2 Run AxoRouter as non-root

Run AxoRouter as a non-root user, and grant operators the commands they need to manage it.

To run AxoRouter as a non-root user, set the `AXO_USER` and `AXO_GROUP` environment variables to the user's username and groupname on the host you want to deploy AxoRouter. For details, see Advanced installation options.

Operators must have access to the following commands:

- `/usr/bin/systemctl * axolet.service` : Controls the `axolet.service` systemd unit. Usually `*` is `start` , `stop` , `restart` , `enable` , and `status` . Used by the operators for troubleshooting.
- `/usr/local/bin/configure-axolet` : Creates initial `axolet` configuration and enables/starts the `axolet` service. Executed by the bootstrap script.
- Command to install and upgrade the `axolet` package. Executed by the bootstrap script if the packages aren't already installed.
  - On RPM-based Linux distributions: `/usr/bin/rpm -Uv axo*.rpm`
  - On DEB-based Linux distributions: `/usr/bin/dpkg -i axo*.deb`
- `/usr/bin/systemctl * axorouter.service` : Controls the `axorouter.service` systemd unit. Usually `*` is `start` , `stop` , `restart` , `enable` , and `status` . Used by the operators for troubleshooting.
- `/usr/local/bin/configure-axorouter` : Creates the initial axorouter configuration and enables/starts the axorouter service. Executed by the bootstrap script.
- Command to install and upgrade the axorouter and the axolet package. Executed by the bootstrap script if the packages aren't already installed.
  - On RPM-based Linux distributions: `/usr/bin/rpm -Uv axo*.rpm`
  - On DEB-based Linux distributions: `/usr/bin/dpkg -i axo*.deb`

You can permit the `syslogng` user to run these commands by running on of the following:

- On RPM-based Linux distributions:

```
sudo tee /etc/sudoers.d/axorouter <<A
axo ALL=(ALL) NOPASSWD: /usr/local/bin/configure-axolet, /usr/local/bin/setup-
axorouter, /usr/bin/rpm -Uv axolet.rpm, /usr/bin/rpm -Uv axorouter.rpm,
/usr/bin/tee /etc/sysctl.d/*, /usr/sbin/sysctl -p /etc/sysctl.d/*
A
```

- On DEB-based Linux distributions: `/usr/bin/dpkg -i axo*.deb`

```
sudo tee /etc/sudoers.d/axorouter <<A
axo ALL=(ALL) NOPASSWD: /usr/local/bin/configure-axolet, /usr/local/bin/setup-
axorouter, /usr/bin/dpkg -i axolet.deb, /usr/bin/dpkg -i axorouter.deb,
/usr/bin/tee /etc/sysctl.d/*, /usr/sbin/sysctl -p /etc/sysctl.d/*
A
```

6.1.1.3 Uninstall AxoRouter from Linux

Stop Axolet, remove the AxoRouter services, packages, and container images, and delete the host from AxoConsole.

To remove AxoRouter from a Linux host, complete the following steps.

1. Log in to the AxoRouter host and stop Axolet first, so it can't modify the AxoRouter configuration and restart services while you remove them:

   ```
   sudo systemctl disable --now axolet
   ```

2. Disable the units that start AxoRouter services automatically, then stop the AxoRouter services:

   ```
   sudo systemctl disable --now axorouter-image-puller.service \
     axorouter-wec.path axostore.path axorouter-resolver.path \
     axorouter-logstore.path axorouter-rsigma.path axoflow-cloudconnector.path
   sudo systemctl stop axorouter axorouter-wec axostore axorouter-resolver \
     axorouter-logstore axorouter-rsigma axoflow-cloudconnector
   ```

   Removing the packages stops the services as well. Stopping them first makes sure that nothing writes to the data directories while you remove them, and that Axolet doesn't register the host again after you delete it from AxoConsole.

3. Delete the host from AxoConsole, so AxoConsole stops managing it.

   1. Open AxoConsole and select **Routers**. Alternatively, select ⌘/Ctrl + K, then enter the name of the host.
   2. Select the name of the host.
   3. Select ⋮ **Host Options > Delete**.
   4. Select **Delete and revoke**. Revoking the host prevents it from registering again with its current identity. If you plan to reinstall on the same host later, select **Delete** to keep the registration.

4. Remove the AxoRouter and Axolet packages:

   - On Red Hat Enterprise Linux 8 and 9, AlmaLinux, and Rocky Linux:

     ```
     sudo dnf remove axorouter axolet
     ```

- On Red Hat Enterprise Linux 7:

```
sudo yum remove axorouter axolet
```

- On Debian and Ubuntu:

```
sudo apt remove axorouter axolet
```

Then reload the systemd configuration, so systemd drops the units of the removed packages:

```
sudo systemctl daemon-reload
```

5. Reclaim the disk space used by the AxoRouter container images:

```
sudo podman system prune -a --volumes
```

> **CAUTION:**
>
> This command removes every unused Podman container, image, and volume on the host, not only the ones of AxoRouter. If you run other Podman workloads on the host, remove the AxoRouter images individually instead (`sudo podman images` , then `sudo podman rmi <image-id>`).

6. (Optional) Delete the data and the configuration of AxoRouter. The packages don't remove these files, so you can reinstall AxoRouter and keep its data.

> **CAUTION:**
>
> Hazard of data loss!
>
> The following command permanently deletes all data stored on the host, including:
>
> - the AxoStore data in `/var/lib/clickhouse` ,
> - the disk buffers of AxoRouter in `/var/lib/syslog-ng` , and

- the log store data (including the `warm` and `archive` directories) in `/var/lib/logstore`. Earlier AxoRouter versions store this data in `/var/lib/syslog-ng/logstore`.

Make sure that you don't need this data before running the command.

If you've set custom locations during the installation (for example, using the `AXO_AXOSTORE_DATA_DIR`, `AXO_PERSIST_MOUNT`, `AXOROUTER_WEC_MOUNT`, `AXO_LOGSTORE_DATA_HOST_DIR`, `AXO_LOGSTORE_WARM_HOST_DIR`, or `AXO_LOGSTORE_ARCHIVE_HOST_DIR` environment variables), delete those directories instead of the defaults.

If you don't plan to reinstall AxoRouter, run:

```
sudo rm -rf /etc/axorouter /etc/axolet \
  /var/lib/syslog-ng /var/lib/clickhouse /var/lib/logstore \
  /var/lib/axorouter /var/lib/axoflow-cloudconnector
```

If you've used custom configuration directories during the installation, the installer has also created systemd drop-in files. Delete them, then reload the systemd configuration:

```
sudo rm -rf /etc/systemd/system/axorouter-logstore.path.d \
  /etc/systemd/system/axorouter-wec.path.d \
  /etc/containers/systemd/axorouter-wec.container.d
sudo systemctl daemon-reload
```

If you've accepted the installer's offer to increase the UDP receive buffer size (`net.core.rmem_max`), delete the setting as well. The original value is restored after the next reboot.

```
sudo rm -f /etc/sysctl.d/99-axolet.conf
```

If you've configured AxoRouter to run as non-root, also delete the sudoers file you've created for it. Depending on which instructions you've followed, the file is called `axorouter`, `axolet`, or `configure-axoflow`:

```
sudo rm -f /etc/sudoers.d/axorouter /etc/sudoers.d/axolet
/etc/sudoers.d/configure-axoflow
```

6.1.2 Install AxoRouter on Kubernetes

Install AxoRouter on a Kubernetes cluster with the one-liner from AxoConsole, create a flow, and send logs to it.

To install AxoRouter on a Kubernetes cluster, complete the following steps. For other platforms, see AxoRouter.

> **Note**
>
> **Note that AxoRouter and Axoflow agent collects detailed, real-time metrics about the data-flows – giving you observability over the health of the security data pipeline and its components. Your security data remains in your self-managed cloud or in your on-prem instance where your sources, destinations, Axoflow agents, and AxoRouters are running, only metrics are forwarded to AxoConsole.**

Prerequisites

Kubernetes version 1.29 and newer

Resource requirements

For a deployment that handles up to 1TB/day log traffic (~14500 EPS) even with complex routing and processing configurations, we recommend:

- 4 vCPU
- 8 GB memory
- 45 GB disk / hour. AxoRouter buffers incoming log data on disk if the destination or the network connection to the destination becomes unavailable. With a 1TB/day throughput, you need at least 45 GB of disk buffer per hour to avoid putting backpressure on your sources. This doesn't include any disk for using AxoStore.

For more details on hardware sizing, contact our support team.

Network access

The host must be able to access the following domains related to the AxoConsole:

- When using AxoConsole SaaS:
  - `<your-tenant-id>.cloud.axoflow.io` : HTTPS traffic on TCP port 443, needed to download the binaries for Axoflow software (like Axolet and AxoRouter).

    - `kcp.<your-tenant-id>.cloud.axoflow.io`: HTTPS (mutual TLS) traffic on TCP port 443 for management traffic.
    - `telemetry.<your-tenant-id>.cloud.axoflow.io`: HTTPS (mutual TLS) traffic on TCP port 443, where Axolet sends the metrics of the host.
    - `us-docker.pkg.dev`: HTTPS traffic on TCP port 443, but only if you're forcing the installation script to pull the container images from the public repository. The Axolet and AxoRouter for Linux installation scripts download the images directly from AxoConsole.
- When using an on-premise AxoConsole:
    - The following domains should point to AxoConsole IP address to access Axoflow from your desktop and AxoRouter hosts:
        - `your-host.your-domain`: The main domain of your AxoConsole deployment.
        - `idp.your-host.your-domain`: A subdomain for the identity provider.
        - `kcp.your-host.your-domain`: A subdomain for management traffic.
        - `telemetry.your-host.your-domain`: A subdomain for agent telemetry.
    - The AxoConsole host must have the following **Open Ports**:
        - 80 TCP: Traefik ingress (HTTP, redirected to HTTPS).
        - 443 TCP: HTTPS for the AxoConsole UI, API, and all web traffic.
- When installing Axoflow agent for Windows or Linux:
    - `github.com`: HTTPS traffic on TCP port 443, for downloading installer packages.

By default, AxoRouter accepts data on the following ports (unless you've modified the default connector rules):

- 514 UDP and TCP for RFC3164 (BSD-syslog) and RFC5424 (IETF-syslog) formatted traffic. AxoRouter automatically recognizes and handles both formats.
- 601 TCP (RFC 3195 - Reliable Delivery for syslog) for RFC5424 (IETF-syslog) and RFC3164 (BSD-syslog) formatted traffic. AxoRouter automatically recognizes and handles both formats.
- 6514 TCP for TLS-encrypted syslog traffic (RFC 5425).
- 4317 TCP for OpenTelemetry log data.
- 5085 TCP for the Windows Event Collector (WEC) service.
- 8088 TCP for the Splunk HEC receiver (optional, only if you forward Splunk HEC traffic to AxoRouter).
- 9200 TCP for the Elasticsearch Bulk API connector (optional, only if you forward data from Elastic agents to AxoRouter).

To receive data on other ports or other protocols, configure other connector rules for the AxoRouter host.

For TLS-encrypted syslog connections, create a new connector rule or edit an existing one, and configure the keys and certificates needed to encrypt the connections. For details, see Syslog.

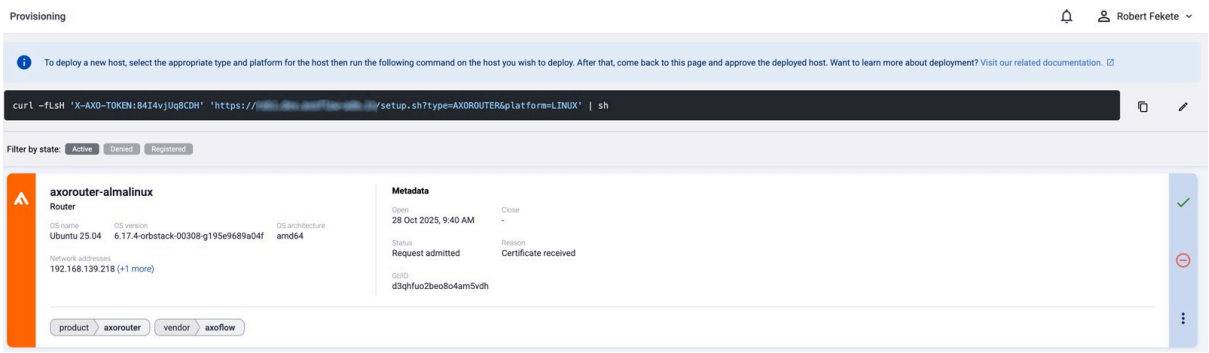
> **Note**
>
> **When you create a connector rule on a new port, open that port in the firewall of of every corresponding AxoRouter as well.**

Install AxoRouter

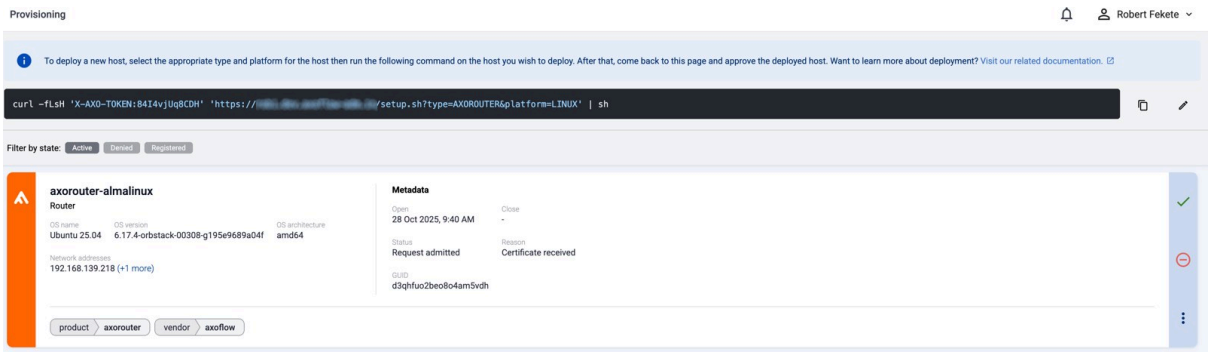
1. Open the AxoConsole.
2. Select **Provisioning**.
3. Select the **Host type > AxoRouter > Kubernetes**. The one-liner installation command is displayed.
4. Open a terminal and set your Kubernetes context to the cluster where you want to install AxoRouter.
5. Run the one-liner, and follow the on-screen instructions.

```
Current kubernetes context: minikube
Server Version: v1.28.3
Installing to new namespace: axorouter
Do you want to install AxoRouter now? [Y]
```

6. Register the host.
    1. Reload the **Provisioning** page. There should be a registration request for the new AxoRouter deployment. Select ✓.



2. Select **Register** to register the host. You can add a description and labels (in `label:value` format) to the host.



3. If the primary IP address (the first IP address shown in the **Network addresses** section on the **Routers** page for each AxoRouter) is not accessible from your edge hosts, set a **Network address override** (IP address or an FQDN) that's accessible. Otherwise, data forwarding from edge hosts will fail.

4. Select the **Topology** page. The new AxoRouter instance is displayed.

Create a flow

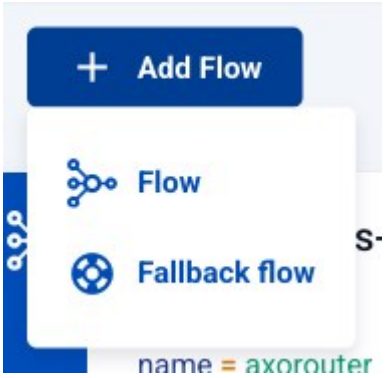
1. If you haven't already done so, create a new destination.

2. **Create a flow to connect the new AxoRouter to the destination.**

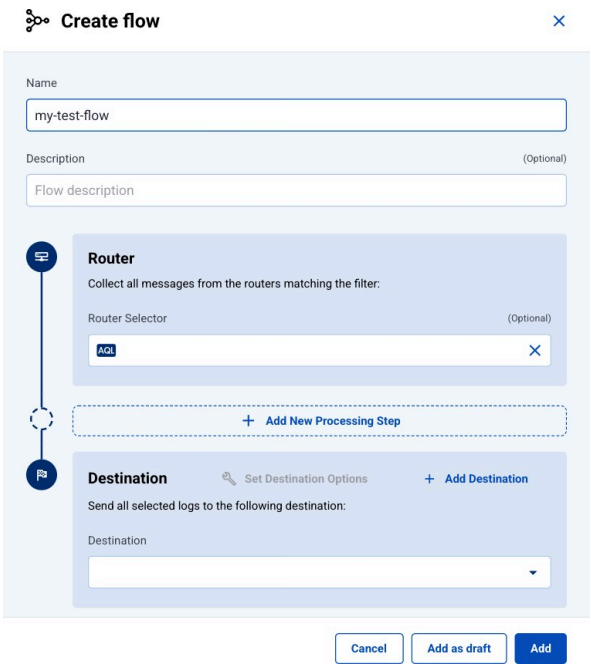
   1. Select **Flows**.

   2. Select **Add Flow > Flow**.

      To create a fallback flow, select **Add Flow > Fallback flow**.



3. Enter a name for the flow, for example, `my-test-flow`.



4. In the **Router Selector** field, enter an expression that matches the routers you want to apply the flow to. To select a specific router, use a name selector, for example, `name = my-axorouter-hostname`.

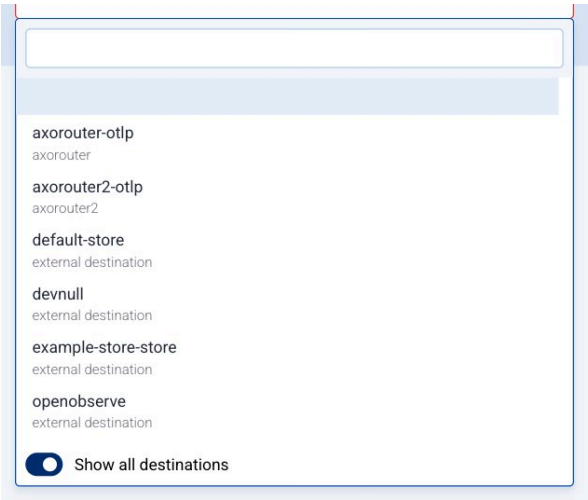
   You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

   - If you leave the router selector empty, it matches every AxoRouter instance.
   - To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.
   - If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

> **Note**
>
> **You can configure multiple fallback flows, but only one fallback flow can apply to an AxoRouter (so the Router Selector of the fallback flows can't overlap).**

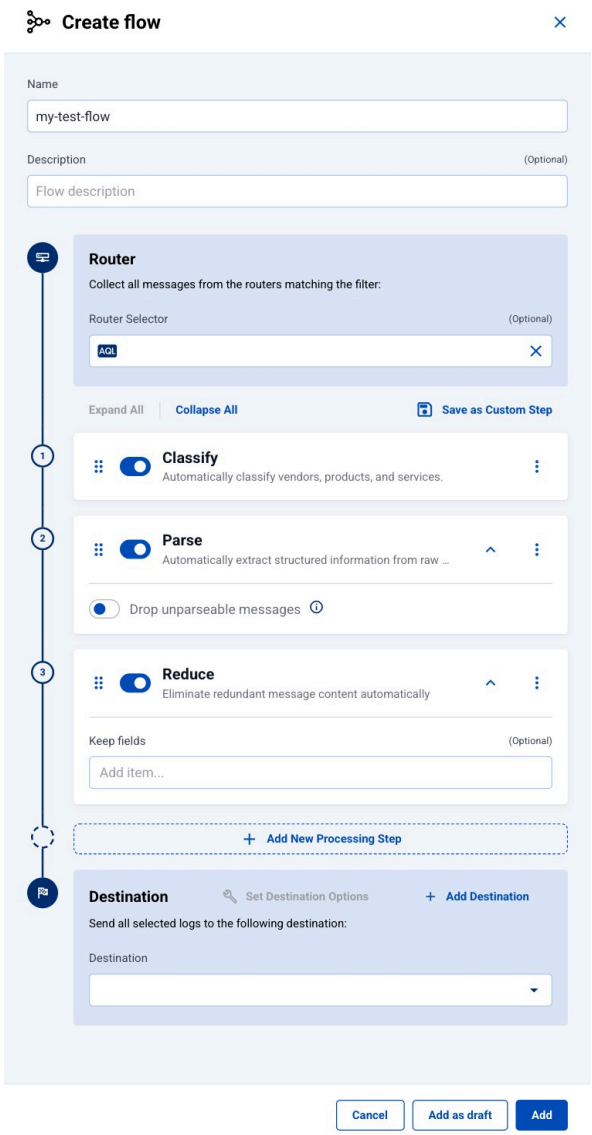
5. Select the **Destination** where you want to send your data. If you don't have any destination configured, you can select **Add Destination** in the destination section to create a new destination now. For details on the different destinations, see Destinations.

   - If you don't have any destination configured, see Destinations.
   - If you've already created a store, it's automatically available as a destination. Note that the **Router Selector** of the flow must match only AxoRouters that have the selected store available, otherwise you'll get an error message.
   - If you want to send data to another AxoRouter, enable the **Show all destinations** option, and select the connector of the AxoRouter where you want to send the data.



6. (Optional) To process the data transferred in the flow, select **Add New Processing Step**. For details, see Processing steps. For example:

   1. Add a **Classify**, a **Parse**, and a **Reduce** step, in that order, to automatically remove redundant and empty fields from your data.
   2. To select which messages are processed by the flow, add a **Select Messages** step, and enter a filter into the **AQL Expression** field. For example, to select only the messages received from Fortinet FortiGate firewalls, use the `meta.vendor = fortinet AND meta.product = fortigate` query.



7. Select **Add**, then confirm with **Add**. To save the flow without provisioning it to the routers yet, select **Add as draft** instead.

8. The new flow appears in the **Flows** list.



Send logs to AxoRouter

By default, AxoRouter accepts data on the following ports (unless you've modified the default connector rules):

- 514 UDP and TCP for RFC3164 (BSD-syslog) and RFC5424 (IETF-syslog) formatted traffic. AxoRouter automatically recognizes and handles both formats.

- 601 TCP (RFC 3195 - Reliable Delivery for syslog) for RFC5424 (IETF-syslog) and RFC3164 (BSD-syslog) formatted traffic. AxoRouter automatically recognizes and handles both formats.
- 6514 TCP for TLS-encrypted syslog traffic (RFC 5425).
- 4317 TCP for OpenTelemetry log data.
- 5085 TCP for the Windows Event Collector (WEC) service.
- 8088 TCP for the Splunk HEC receiver (optional, only if you forward Splunk HEC traffic to AxoRouter).
- 9200 TCP for the Elasticsearch Bulk API connector (optional, only if you forward data from Elastic agents to AxoRouter).

To receive data on other ports or other protocols, configure other connector rules for the AxoRouter host.

For TLS-encrypted syslog connections, create a new connector rule or edit an existing one, and configure the keys and certificates needed to encrypt the connections. For details, see Syslog.

> **Note**
>
> **When you create a connector rule on a new port, open that port in the firewall of of every corresponding AxoRouter as well.**

Upgrade AxoRouter

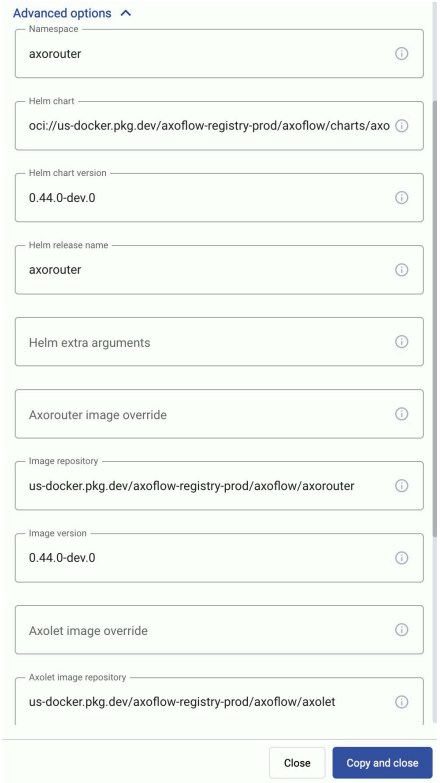
AxoConsole raises an alert for the host when a new AxoRouter version is available. To upgrade to the new version, re-run the one-liner installation command you used to install AxoRouter, or select **Provisioning > Select type and platform** to create a new one.

Note that it can take a few minutes until the version numbers of the different services get updated on AxoConsole.

6.1.2.1 Advanced installation options

Proxy settings, Helm chart and image options, and Axolet parameters for installing AxoRouter on Kubernetes.

When installing AxoRouter, you can set a number of advanced options if needed for your environment. Setting the advanced options in the AxoConsole automatically updates the one-liner command that you can copy and run.



Alternatively, before running the one-liner you can use one of the following methods:

- Set the related environment variable for the option. For example:

```
export AXOROUTER_USER=syslogng
export AXOROUTER_GROUP=syslogng
```

- Set the related URL parameter for the option. For example:

```
curl -fLsH 'X-AXO-TOKEN:random-generated' 'https://<your-tenant-
id>.cloud.axoflow.io/setup.sh?type=AXOROUTER&platform=K8S&parameter=value' | sh
```

Proxy settings

Use the `http_proxy=` , `https_proxy=` , `no_proxy=` parameters to configure HTTP proxy settings for the installer. To configure the Axolet service to use the proxy settings, enable the `AXOLET_AVOID_PROXY` parameter as well. Lowercase variable names are preferred because they work universally.

Installation options

You can pass the following parameters to the installation script as environment variables, or as URL parameters.

> **Note**
>
> **Running the provisioning command with `sudo` would mask environment variables of the calling shell. Either start the whole procedure from a root shell, or let the install script call sudo when it needs to. In other words: don't add the `sudo` command to the provisioning command.**

AxoRouter image override

Default value:	empty string
Environment variable	`IMAGE`
URL parameter	`image`

Description: Deploy the specified AxoRouter image.

Helm chart

Default value:	`oci://us-docker.pkg.dev/axoflow-registry-prod/axoflow/charts/axorouter-syslog`
Environment variable	`HELM_CHART`
URL parameter	`helm_chart`

Description: The path or URL of the AxoRouter Helm chart.

Helm chart version

Default value:	Current Axoflow version
Environment variable	`HELM_CHART_VERSION`
URL parameter	`helm_chart_version`

Description: Deploy the specified version of the Helm chart.

Helm extra arguments

Default value:	empty string
Environment variable	`HELM_EXTRA_ARGS`
URL parameter	`helm_extra_args`

Description: Additional arguments passed to Helm during the installation.

Helm release name

Default value:	`axorouter`
Environment variable	`HELM_RELEASE_NAME`
URL parameter	`helm_release_name`

Description: Name of the Helm release.

Image repository

Default value:	`us-docker.pkg.dev/axoflow-registry-prod/axoflow/axorouter`
Environment variable	`IMAGE_REPO`
URL parameter	`image_repo`

Description: Deploy AxoRouter from a custom image repository.

Image version

Default value:	Current Axoflow version
Environment variable	`IMAGE_VERSION`
URL parameter	`image_version`

Description: Deploy the specified AxoRouter version.

Namespace

Default value:	`axorouter`
Environment variable	`NAMESPACE`
URL parameter	`namespace`

Description: The namespace where AxoRouter is installed.

Axolet parameters

API server host

Default value:	
Environment variable	
URL parameter	`api_server_host`

Description: Override the host part of the API endpoint for the host.

Axolet executable path

Default value:	
Environment variable	`AXOLET_EXECUTABLE`
URL parameter	`axolet_executable`

Description: Path to the Axolet executable.

Axolet image override

Default value:	empty string
Environment variable	`AXOLET_IMAGE`
URL parameter	`axolet_image`

Description: Deploy the specified Axolet image.

Axolet image repository

Default value:	`us-docker.pkg.dev/axoflow-registry-prod/axoflow/axolet`
Environment variable	`AXOLET_IMAGE_REPO`
URL parameter	`axolet_image_repo`

Description: Deploy Axolet from a custom image repository.

Axolet image version

Default value:	Current Axoflow version
Environment variable	`AXOLET_IMAGE_VERSION`
URL parameter	`axolet_image_version`

Description: Deploy the specified Axolet version.

Initial GUID

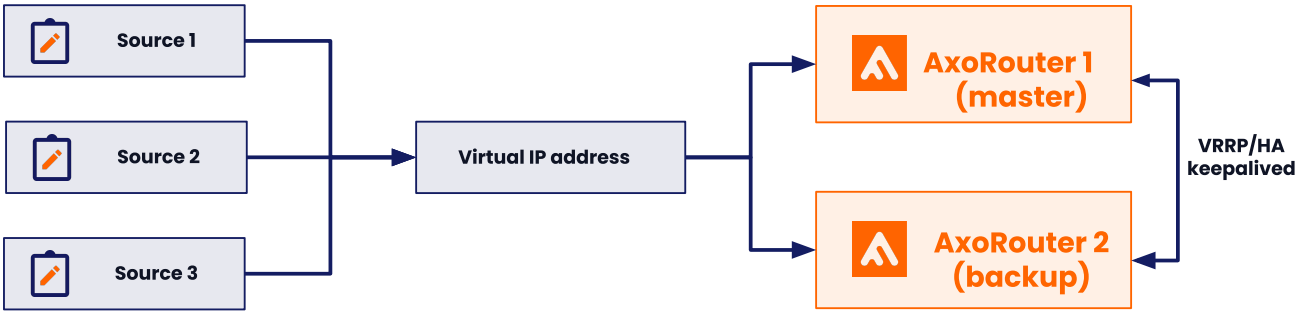
Default value:	
Environment variable	
URL parameter	`initial_guid`

Description: Set a static GUID.

6.1.3 Set up AxoRouter high availability with keepalived

Deploy AxoRouter as a highly available cluster using keepalived and a VRRP-managed virtual IP that fails over between primary and backup nodes without a load balancer.

This guide describes how to deploy multiple AxoRouter hosts as a high availability cluster with `keepalived` managing a shared virtual IP across the hosts. Instead of relying on a dedicated load balancer, `keepalived` uses VRRP to keep the virtual IP on a healthy node and fail it over automatically, so this setup needs no external load-balancing hardware. For an overview of other approaches, including putting a load balancer in front of the cluster, see AxoRouter high availability deployment options.



The examples in this guide deploy AxoRouter on AlmaLinux 10, but the procedure also works on other supported platforms.

The hosts running AxoRouter can be physical or virtual machines. The recommended layout depends on which you choose:

- Physical machines: Install two machines with a dedicated HA link between them to keep the HA traffic separate from the production traffic.
- Virtual machines: Set up three VMs with a dedicated (and guaranteed bandwidth) HA link (HA VLAN) spanning across them, to keep the HA traffic separate from the production traffic.

> **Note**
>
> A cluster targeting highly reliable services should ideally consist of at least three nodes to avoid split-brain situations. In a split-brain situation, both nodes think that they are the last surviving node (for example, during a network failure) and bring up their services, leading to several problems.
>
> The setups described here use fewer than the ideal number of nodes for cost reasons. Dedicated HA links provide an additional channel for HA communication independently of the production traffic, and significantly reduce the risk of split-brain scenarios.

At a high level, setting up the cluster involves the following steps:

1. Prepare every cluster host: deploy AxoRouter, install `keepalived`, and configure SELinux to let `keepalived` query the AxoRouter service status.
2. Configure `keepalived` on the primary node with the `MASTER` state, the shared virtual IP, and the addresses of the peer nodes.
3. Configure `keepalived` on each secondary node with the `BACKUP` state and a lower priority than the primary.
4. Open TCP port 112 on every host's firewall so the nodes can exchange VRRP advertisements, then enable and start the `keepalived` service.

The following sections walk through each step in detail.

Prepare the hosts and install keepalived

On every AxoRouter host that is a member of the high availability cluster, complete the following steps.

1. Deploy AxoRouter on every host as described in Install AxoRouter on Linux.
2. Install `keepalived` from the OS repositories.

```
sudo dnf install keepalived
```

3. Configure SELinux so `keepalived` can monitor the status of AxoRouter. For that, create and load a SELinux policy module that allows `keepalived` to run the `systemctl status` command.
    1. Install the `selinux-policy-devel` package:

```
sudo dnf install selinux-policy-devel
```

    2. Create a policy module file.

```
cat > keepalived-allow-systemctl-status.te <<EOF

module keepalived-allow-systemctl-status 1.0;

require {
        type keepalived_t;
        type syslogd_var_run_t;
        type systemd_systemctl_exec_t;
        type systemd_generator_unit_file_t;
        class file { execute execute_no_trans getattr map open read setattr
};
        class dir read;
        class service status;
}

#============= keepalived_t ==============
allow keepalived_t syslogd_var_run_t:dir read;

#!!!! This avc can be allowed using the boolean 'domain_can_mmap_files'
allow keepalived_t systemd_systemctl_exec_t:file { execute execute_no_trans
getattr open read setattr };

#!!!! This avc can be allowed using the boolean 'domain_can_mmap_files'
allow keepalived_t systemd_systemctl_exec_t:file map;

allow keepalived_t systemd_generator_unit_file_t:service status;
EOF
```

3. Build and load the SELinux module:

```
checkmodule -M -m -o keepalived-allow-systemctl-status.mod keepalived-
allow-systemctl-status.te
semodule_package -o keepalived-allow-systemctl-status.pp -m keepalived-
allow-systemctl-status.mod
sudo semodule -i keepalived-allow-systemctl-status.pp
```

4. Repeat these steps on the other AxoRouter hosts.

Configure keepalived as MASTER (primary node)

1. Set up the `keepalived` configuration on the primary node. See the list of main configuration elements after the command.

```
sudo mv /etc/keepalived/keepalived.conf{,.old} -v
sudo cat > /etc/keepalived/keepalived.conf <<EOF

vrrp_script check_axorouter {
    script  "/usr/bin/systemctl status axorouter"
    interval 5
    timeout  2
    rise     5
    fall     2
    user     root
}

vrrp_instance axorouter_vip {
    state MASTER
    interface enp0s8
    virtual_router_id 51
    priority 254
    advert_int 1
    unicast_src_ip 192.168.56.10
    unicast_peer {
        192.168.56.11
    }
    authentication {
        auth_type PASS
        auth_pass Ax0R0ut3r
    }
    virtual_ipaddress {
        192.168.10.64/25 dev enp0s9
    }
    track_script {
        check_axorouter
    }
    track_interface {
        enp0s9                      # Fail over if public interface dies
        enp0s8                      # Fail over if dedicated HA interface dies
    }
}
EOF
```

The main configuration elements are:

- `state` : The default state of the node.
- `interface` : The interface `keepalived` is listening on (in this case the HA interface).
- `virtual_router_id` : This ID should match across the nodes belonging to the same cluster.
- `priority` : An assigned value belonging to the cluster node. During an election, the node with the highest priority wins. The maximum value is 254.
- `unicast_src_ip` : The source IP address `keepalived` sends its advertisements with (in this case on the HA interface).
- `unicast_peer` : IP addresses of the other cluster nodes (in this case the IP addresses on their respective HA interfaces).
- `auth_pass` : The password can have any value. Maximum length is 8 characters; longer values are truncated. All members of the cluster must have the same value.
- `virtual_ipaddress` : The virtual IP address and the interface it should be configured on (the interface facing the data sources).
- `track_script` : A shell script that can be used to verify the status of the service.
- `track_interface` : A list of network interfaces the status of which `keepalived` should monitor (the interface `keepalived` is listening on is in this list by default).

2. Enable inbound traffic on port 112/TCP on the host's firewall.

```
sudo firewall-cmd --permanent --add-port=112/tcp && firewall-cmd --reload
```

3. Enable and start the `keepalived` service.

```
sudo systemctl enable --now keepalived
```

Configure keepalived as BACKUP (secondary nodes)

Perform the following steps on the secondary hosts.

1. Deploy the following `keepalived` configuration.

```
sudo mv /etc/keepalived/keepalived.conf{,.old} -v
sudo cat > /etc/keepalived/keepalived.conf <<EOF

vrrp_script check_axorouter {
    script  "/usr/bin/systemctl status axorouter"
    interval 5
    timeout  2
    rise     5
    fall     2
    user     root
}

vrrp_instance axorouter_vip {
    state BACKUP
    interface enp0s8
    virtual_router_id 51
    priority 253
    advert_int 1
    unicast_src_ip 192.168.56.11
    unicast_peer {
        192.168.56.10
    }
    authentication {
        auth_type PASS
        auth_pass Ax0R0ut3r
    }
    virtual_ipaddress {
        192.168.10.64/25 dev enp0s9
    }
    track_script {
        check_axorouter
    }
    track_interface {
        enp0s9                      # Fail over if public interface dies
        enp0s8                      # Fail over if dedicated HA interface dies
    }
}
EOF
```

2. Enable inbound traffic at port 112/TCP on the host's firewall.

```
sudo firewall-cmd --permanent --add-port=112/tcp && firewall-cmd --reload
```

3. Enable and start the `keepalived` service.

```
sudo systemctl enable --now keepalived
```

After this, you should have a functional virtual/service IP address so the virtual IP moves to another AxoRouter host if the active one fails.

6.2 AxoSyslog

Install Axolet on a host that runs AxoSyslog, and configure AxoSyslog so that AxoConsole receives its metrics.

Onboarding allows you to collect metrics about the host, display the host on the **Topology** page, and to tap into the log flow.

Onboarding requires you to modify the host and the configuration of the logging agent (AxoSyslog) running on the host.

- Level 1: Install Axolet on the host where AxoSyslog is running. Axolet collects metrics from the host and sends them to the AxoConsole, so you can check host-level metrics on the **Metrics & Health** page of the host, and displays the host on the **Topology** page.
- Level 2: Instrument the configuration of the logging agent to provide detailed metrics about the traffic flow. This allows you to display data about the host on the **Analytics** page.
- Level 3: Instrument the configuration of the logging agent to allow you to access the logs of the logging agent and to tap into the log flow from the AxoConsole. The exact steps for this integration depend on the configuration of your logging agent. We provide basic instrumentation instructions for getting started in this guide, but we strongly recommend to contact us so our professional services can help you with a production integration.

To onboard an existing AxoSyslog instance into Axoflow, complete the following steps.

1. Install Axolet on the host, then approve its registration on the **Provisioning** page of the AxoConsole.
2. The AxoSyslog host is now visible on the **Topology** page of the AxoConsole as a source.
3. Access the AxoSyslog host and edit the configuration of AxoSyslog. Set the statistics-related global options like this (if the `options` block already exists, add these lines to the bottom of the block):

```
options {
    stats(
        level(2)
        freq(0) # Inhibit statistics output to stdout
        );
};
```

4. (Optional) To get detailed metrics and analytics about the traffic that flows through the host, instrument your AxoSyslog configuration as follows:

> **Note**
>
> **You can use Axolet with an un-instrumented AxoSyslog configuration file, but that limits available metrics to host statistics (for example, disk, memory, queue information). You won't be able to access data about the actual traffic flowing through the host. To collect traffic-related metrics, instrument configuration with `metrics-probe()` stanzas. The example below shows how to instrument the configuration to highlight common macros such as `$HOST` and `$PROTOCOL`. If you want to customize the collected metrics or need help with the instrumentation, contact us.**

1. Download the following configuration snippet to the AxoSyslog host, for example, as `/etc/syslog-ng/conf.d/axoflow-instrumentation.conf` for AxoSyslog.

```
# Metrics probe parser definition for a specified destination
# This will yield a reasonable number of metrics for common
# syslog-ng macros
block parser metrics-output(destination(undefined)) {

  # This will probe for the number of *events*.
  # key name is fixed; labels are user-specified
  metrics-probe(
    key("classified_output_events_total")
    labels(
      "host" => "${HOST}"
      "service" => "${PROGRAM}"
      "source" => "${SOURCE}"
      "dest_port" => "$DESTPORT"
      "dest_ip" => "$DESTIP"
      "src_ip" => "$SOURCEIP"
      "protocol" => "$(if ($PROTO == 6) TCP $(if ($PROTO == 17) UDP
$PROTO))"
      "ip_protocol" => "$IP_PROTO"
      "transport" => "$TRANSPORT"
      "destination_name" => "`destination`"
      "issue" => "$(or $(tags-head message.parse_error
syslog.unexpected_framing syslog.missing_pri syslog.rfc3164_missing_header
syslog.invalid_hostname message.utf8_sanitized) 'none')"
```

```
    )
  );

  # This will probe for the number of *bytes*.
  # key name is fixed; labels are user-specified
  metrics-probe(
    key("classified_output_event_bytes_total")
    increment("${RAWMSG_SIZE}")
    labels(
      "host" => "${HOST}"
      "service" => "${PROGRAM}"
      "source" => "${SOURCE}"
      "dest_port" => "$DESTPORT"
      "dest_ip" => "$DESTIP"
      "src_ip" => "$SOURCEIP"
      "protocol" => "$(if ($PROTO == 6) TCP $(if ($PROTO == 17) UDP
$PROTO))"
      "ip_protocol" => "$IP_PROTO"
      "transport" => "$TRANSPORT"
      "destination_name" => "`destination`"
      "issue" => "$(or $(tags-head message.parse_error
syslog.unexpected_framing syslog.missing_pri syslog.rfc3164_missing_header
syslog.invalid_hostname message.utf8_sanitized) 'none')"
    )
  );
};


@define AXOTAP_SOCK "/var/lib/syslog-ng/axotap.sock"

block rewrite format_axotap_sock_message(service_type("input")) {
  set("$(substr ${MESSAGE} 0 32768)" value(MESSAGE));
  set("$(substr ${RAWMSG} 0 32768)" value(RAWMSG));
  set("$(format-json --no-cast debug.msg=$(format-json --no-cast --scope
all-nv-pairs) log.body=${MESSAGE})" value(".tap.event"));
  set(json('{"note":"message object too long for tapping"}')
value(".tap.event") condition("$(length ${.tap.event})" > 131072));

  set("$(format-json --no-cast tap=$(format-json --no-cast tapId=''
serviceType='`service_type`' eventId=${UNIQID} labels=$(format-json --no-
cast) hostCandidate=$(format-json --no-cast) facility=${FACILITY}
format=${MSGFORMAT} hostFrom=${HOST_FROM} message=$(echo ${MESSAGE})
```

```
rawMessage=${RAWMSG}severity=${SEVERITY}source=${SOURCE}tags=${TAGS}timesta
mp=${ISODATE}transport=${TRANSPORT})event=json(${.tap.event}))"
value(".tap.formatted"));};


# input side tapping
destination "d_axotap_input_static" {
  channel {
    filter {
      rate-limit(key("$SOURCE:$PROGRAM:$HOST:$SOURCEIP:$DESTIP") rate(1));
    };

    rewrite {
      format_axotap_sock_message(service_type("input"));
    };

    destination {
      unix-dgram("`AXOTAP_SOCK`" persist-name("axotap_input_static")
template("${.tap.formatted}"));
    };
  };
};


log {
  destination("d_axotap_input_static");
  flags(catchall);
};


# output side tapping
destination "d_axotap_output" {
  channel {
    filter {
      rate-limit(key("$SOURCE:$PROGRAM:$HOST:$SOURCEIP:$DESTIP") rate(1));
    };

    rewrite {
      format_axotap_sock_message(service_type("output"));
    };
```

```
    destination {
      unix-dgram("`AXOTAP_SOCK`" template("${.tap.formatted}"));
    };
  };
};
```

2. Include it in at the top of your configuration file:

```
@version: current
@include "axoflow-instrumentation.conf"
```

3. Edit every destination statement to include a `parser { metrics-output(destination(my-destination)); };` line (making sure to include the `channel` construct if your destination block does not already contain it):

```
destination my-destination {
    channel {
        parser { metrics-output(destination(my-destination)); };
        destination { file("/dev/null"); };
        };
    };
```

4. Reload the configuration of AxoSyslog.

```
systemctl reload syslog-ng
```

5. To enable log tapping for the output logs (traffic that the host sends to its destinations), add the `d_axotap_output` destination in your log paths. This allows the Axolet agent to collect rate-limited log samples from that specific point in the pipeline.

   Tapping the input logs (traffic arriving to the sources of the host) needs only the configuration snippet.

> **Note**
>
> **There are multiple ways to add tapping to your configuration, which in some cases can introduce unwanted side effects. Contact us to help you create a safe tapping configuration for your use case!**

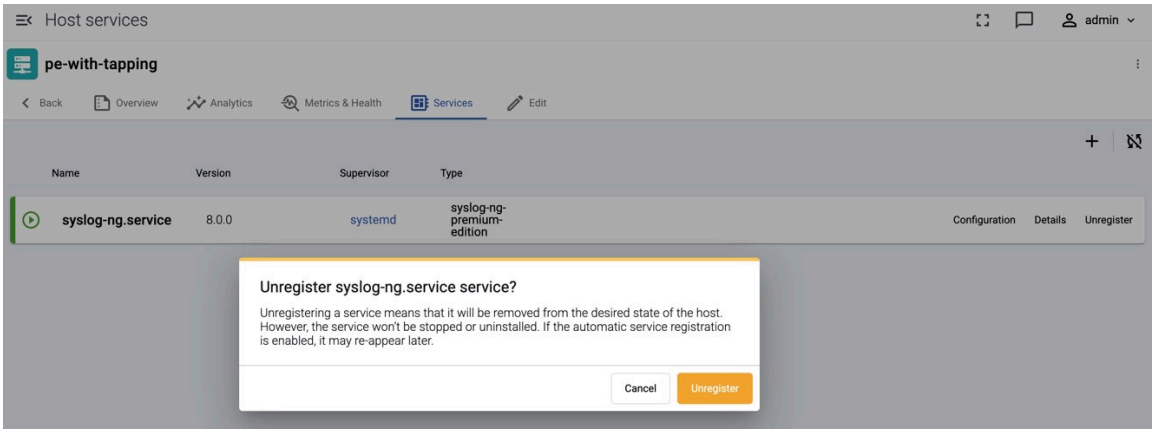
1. Find a log path that you want to tap into and add the tap destination in a safe way, for example, by adding it in a separate inner log path:

```
log {
    source(s_udp514);
    log { destination(d_udp514); };
    log { destination(d_axotap); };
};
```

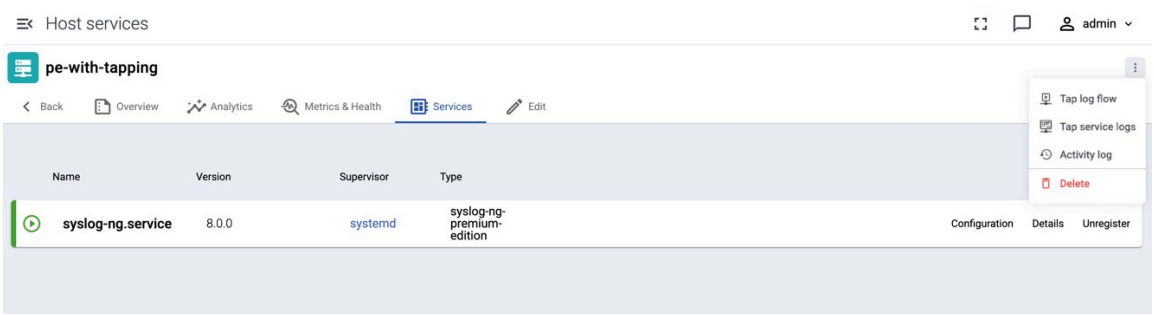
2. Reload the configuration of AxoSyslog.

```
systemctl reload syslog-ng
```

3. Unregister the service so that auto service registration can discover the new tapping point:



4. Test log tapping from the top right menu:



5. If AxoSyslog is sending data to an AxoRouter, create a placeholder data forwarding rule where the **Edge Selector** matches AxoSyslog and the **Router connector** is the connector of the AxoRouter instance where AxoSyslog is sending the data. This is needed to display AxoSyslog and its data metrics on the **Topology** page, AxoConsole won't manage AxoSyslog.

6.3 Splunk Connect for Syslog (SC4S)

Install Axolet on a host that runs Splunk Connect for Syslog (SC4S), and configure SC4S so that AxoConsole receives its metrics.

Onboarding allows you to collect metrics about the host, display the host on the **Topology** page, and to tap into the log flow.

Onboarding requires you to modify the host and the configuration of the logging agent (SC4S) running on the host.

- Level 1: Install Axolet on the host where SC4S is running. Axolet collects metrics from the host and sends them to the AxoConsole, so you can check host-level metrics on the **Metrics & Health** page of the host, and displays the host on the **Topology** page.
- Level 2: Instrument the configuration of the logging agent to provide detailed metrics about the traffic flow. This allows you to display data about the host on the **Analytics** page.
- Level 3: Instrument the configuration of the logging agent to allow you to access the logs of the logging agent and to tap into the log flow from the AxoConsole. The exact steps for this integration depend on the configuration of your logging agent. We provide basic instrumentation instructions for getting started in this guide, but we strongly recommend to contact us so our professional services can help you with a production integration.

To generate metrics for the Axoflow platform from an existing Splunk Connect for Syslog (SC4S) instance, you need to configure SC4S to generate these metrics. Complete the following steps.

1. If you haven't already done so, install Axolet on the host, then approve its registration on the **Provisioning** page of the AxoConsole.
2. Download the following code snippet as `axoflow-instrumentation.conf`.

```
@define allow-config-dups 1

block parser classifier-parser() {
  metrics-probe(
    key("classified_events_total")
    labels(
      "app" => "${.splunk.sourcetype}"
      "host" => "${HOST}"
      "program" => "${.splunk.source}"
      "source" => "${SOURCE}"
    )
  );
};

@module confgen context(root) name(dest_hec)
exec("`SC4S_ETC`/conf.d/destinations/dest_hec/plugin.py | sed -e '/^destination
d_hec/a\    channel { parser { classifier-parser(); }; destination {' -e
'/^};/i\ };};'  ")
dest_hec()

@define allow-config-dups 0
```

3. If you are running SC4S under podman or docker, copy the file into the `/opt/sc4s/local/config/destinations` directory. In other deployment methods this might be different, check the SC4S documentation for details.

4. Check if the metrics are appearing, for example, run the following command on the SC4S host:

```
syslog-ng-ctl stats prometheus | grep classified
```

5. If SC4S is sending data to an AxoRouter, create a placeholder data forwarding rule where the **Edge Selector** matches SC4S and the **Router connector** is the connector of the AxoRouter instance where SC4S is sending the data. This is needed to display SC4S and its data metrics on the **Topology** page, AxoConsole won't manage SC4S.

6.4 syslog-ng

Install Axolet on a host that runs syslog-ng, and configure syslog-ng so that AxoConsole receives its metrics.

Onboarding allows you to collect metrics about the host, display the host on the **Topology** page, and to tap into the log flow.

Onboarding requires you to modify the host and the configuration of the logging agent (syslog-ng) running on the host.

- Level 1: Install Axolet on the host where syslog-ng is running. Axolet collects metrics from the host and sends them to the AxoConsole, so you can check host-level metrics on the **Metrics & Health** page of the host, and displays the host on the **Topology** page.
- Level 2: Instrument the configuration of the logging agent to provide detailed metrics about the traffic flow. This allows you to display data about the host on the **Analytics** page.
- Level 3: Instrument the configuration of the logging agent to allow you to access the logs of the logging agent and to tap into the log flow from the AxoConsole. The exact steps for this integration depend on the configuration of your logging agent. We provide basic instrumentation instructions for getting started in this guide, but we strongly recommend to contact us so our professional services can help you with a production integration.

Prerequisites

For syslog-ng PE, version 7.0.33 or newer is required. Note that the 7.0.x product line needs a different configuration snippet than the 8.1.x product line.

For syslog-ng OSE, version 4.1.1 or newer is required, but we recommend using AxoSyslog instead (at least syslog-ng OSE version 4.5.0).

Provision syslog-ng

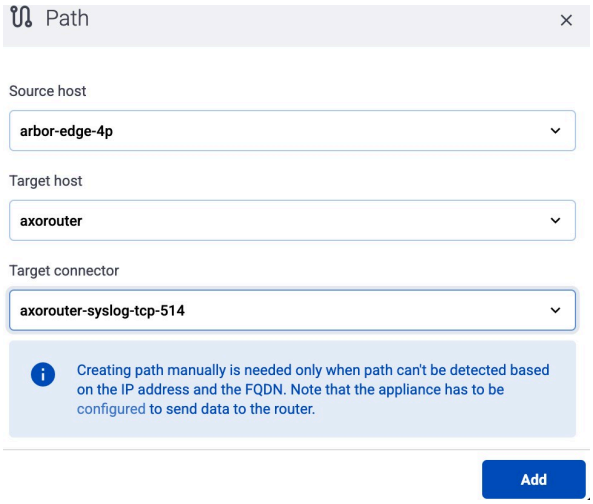
To onboard an existing syslog-ng instance into Axoflow, complete the following steps.

1. Install Axolet on the host, then approve its registration on the **Provisioning** page of the AxoConsole.

2. The syslog-ng host will now be visible on the **Topology** page of the AxoConsole as a source.

3. **If you've already added the AxoRouter instance or other destination where this newly-registered host is sending data to the AxoConsole, add a path to connect the host to the AxoRouter or the destination.**

    1. Select **Topology > Add Item > Path**.



    2. If not set automatically, select the beginning of the path in the **Source host** field.
    3. Select the target router or aggregator this source is sending its data to in the **Target host** field, for example, `axorouter`.
    4. Select the **Target connector**. The connector determines how the destination receives the data (for example, using which protocol or port).
    5. Select **Add**. The new path appears on the **Topology** page.



4. Access the syslog-ng host and edit the configuration of syslog-ng. Set the statistics-related global options like this (if the `options` block already exists, add these lines to the bottom of the block):

```
options {
    stats-level(2);
    stats-freq(0); # Inhibit statistics output to stdout
};
```

5. (Optional) To get detailed metrics and analytics about the traffic that flows through the host, instrument your syslog-ng configuration as follows:

> **Note**
>
> **You can use Axolet with an un-instrumented syslog-ng configuration file, but that limits available metrics to host statistics (for example, disk, memory, queue information). You won't be able to access data about the actual traffic flowing through the host. To collect traffic-related metrics, instrument configuration with `metrics-probe()` stanzas. The example below shows how to instrument the configuration to highlight common macros such as `$HOST` and `$PROTOCOL`. If you want to customize the collected metrics or need help with the instrumentation, contact us.**

1. Download the following configuration snippet to the syslog-ng host, for example, as `/etc/syslog-ng/conf.d/axoflow-instrumentation.conf` for syslog-ng.

   For syslog-ng PE, use `/opt/syslog-ng/etc/axoflow-instrumentation.conf`.

   > **For syslog-ng PE 8.x and newer:**

```
# Metrics probe parser definition for a specified destination
# This will yield a reasonable number of metrics for common
# syslog-ng macros
block parser metrics-output(destination(undefined)) {

  # This will probe for the number of *events*.
  # key name is fixed; labels are user-specified
  metrics-probe(
    key("classified_output_events_total")
    labels(
      "host" => "${HOST}"
      "service" => "${PROGRAM}"
      "source" => "${SOURCE}"
      "dest_port" => "$DESTPORT"
      "dest_ip" => "$DESTIP"
      "src_ip" => "$SOURCEIP"
      "protocol" => "$(if ($PROTO == 6) TCP $(if ($PROTO == 17) UDP
$PROTO))"
      "ip_protocol" => "$IP_PROTO"
      "transport" => "$TRANSPORT"
      "destination_name" => "`destination`"
      "issue" => "$(or $(tags-head message.parse_error
syslog.unexpected_framing syslog.missing_pri
syslog.rfc3164_missing_header syslog.invalid_hostname
message.utf8_sanitized) 'none')"
```

```
    )
  );

  # This will probe for the number of *bytes*.
  # key name is fixed; labels are user-specified
  metrics-probe(
    key("classified_output_event_bytes_total")
    increment("${RAWMSG_SIZE}")
    labels(
        "host" => "${HOST}"
        "service" => "${PROGRAM}"
        "source" => "${SOURCE}"
        "dest_port" => "$DESTPORT"
        "dest_ip" => "$DESTIP"
        "src_ip" => "$SOURCEIP"
        "protocol" => "$(if ($PROTO == 6) TCP $(if ($PROTO == 17) UDP
$PROTO))"
        "ip_protocol" => "$IP_PROTO"
        "transport" => "$TRANSPORT"
        "destination_name" => "`destination`"
        "issue" => "$(or $(tags-head message.parse_error
syslog.unexpected_framing syslog.missing_pri
syslog.rfc3164_missing_header syslog.invalid_hostname
message.utf8_sanitized) 'none')"
    )
  );
};


# For syslog-ng OSE this should be /var/lib/syslog-ng/axotap.sock
@define AXOTAP_SOCK "/opt/syslog-ng/var/axotap.sock"

block rewrite format_axotap_sock_message(service_type("input")) {
  set("$(substr ${MESSAGE} 0 32768)" value(MESSAGE));
  set("$(substr ${RAWMSG} 0 32768)" value(RAWMSG));
  set("$(format-json --no-cast debug.msg=$(format-json --no-cast --scope
all-nv-pairs) log.body=${MESSAGE})" value(".tap.event"));
  set(json('{"note":"message object too long for tapping"}')
value(".tap.event") condition("$(length ${.tap.event})" > 131072));

  set("$(format-json --no-cast tap=$(format-json --no-cast tapId=''
serviceType='`service_type`' eventId=${UNIQID} labels=$(format-json --no-
cast) hostCandidate=$(format-json --no-cast) facility=${FACILITY}
```

```
format=${MSGFORMAT}hostFrom=${HOST_FROM}message=$(echo${MESSAGE})rawMessa
ge=${RAWMSG}severity=${SEVERITY}source=${SOURCE}tags=${TAGS}timestamp=${I
SODATE}transport=${TRANSPORT})event=json(${.tap.event}))"
value(".tap.formatted"));};


# input side tapping
destination "d_axotap_input_static" {
  channel {
    filter {
      rate-limit(key("$SOURCE:$PROGRAM:$HOST:$SOURCEIP:$DESTIP")
rate(1));
    };

    rewrite {
      format_axotap_sock_message(service_type("input"));
    };

    destination {
      unix-dgram("`AXOTAP_SOCK`" persist-name("axotap_input_static")
template("${.tap.formatted}"));
    };
  };
};


log {
  destination("d_axotap_input_static");
  flags(catchall);
};


# output side tapping
destination "d_axotap_output" {
  channel {
    filter {
      rate-limit(key("$SOURCE:$PROGRAM:$HOST:$SOURCEIP:$DESTIP")
rate(1));
    };
```

```
    rewrite {
      format_axotap_sock_message(service_type("output"));
    };

    destination {
      unix-dgram("`AXOTAP_SOCK`" template("${.tap.formatted}"));
    };
  };
};
```

For syslog-ng PE 7.0.33 and newer 7.0.x, and syslog-ng OSE version 4.1.1-4.4.0:

```
# Metrics probe parser definition for a specified destination
# This will yield a reasonable number of metrics for common
# syslog-ng macros
block parser metrics-output(destination(undefined)) {

  # This will probe for the number of *events*.
  # key name is fixed; labels are user-specified
  metrics-probe(
    key("classified_output_events_total")
    labels(
      "host" => "${HOST}"
      "service" => "${PROGRAM}"
      "source" => "${SOURCE}"
      "dest_port" => "$DESTPORT"
      "dest_ip" => "$DESTIP"
      "src_ip" => "$SOURCEIP"
      "protocol" => "$(if ($PROTO == 6) TCP $(if ($PROTO == 17) UDP
$PROTO))"
      "ip_protocol" => "$IP_PROTO"
      "transport" => "$TRANSPORT"
      "destination_name" => "`destination`"
    )
  );

  # This will probe for the number of *bytes*.
  # key name is fixed; labels are user-specified
  metrics-probe(
    key("classified_output_event_bytes_total")
    increment("${RAWMSG_SIZE}")
```

```
    labels(
        "host" => "${HOST}"
        "service" => "${PROGRAM}"
        "source" => "${SOURCE}"
        "dest_port" => "$DESTPORT"
        "dest_ip" => "$DESTIP"
        "src_ip" => "$SOURCEIP"
        "protocol" => "$(if ($PROTO == 6) TCP $(if ($PROTO == 17) UDP
$PROTO))"
        "ip_protocol" => "$IP_PROTO"
        "transport" => "$TRANSPORT"
        "destination_name" => "`destination`"
    )
  );
};


# For syslog-ng OSE this should be /var/lib/syslog-ng/axotap.sock
@define AXOTAP_SOCK "/opt/syslog-ng/var/run/axotap.sock"

block rewrite format_axotap_sock_message(service_type("input")) {
  set("$(substr ${MESSAGE} 0 32768)" value(MESSAGE));
  set("$(substr ${RAWMSG} 0 32768)" value(RAWMSG));
  set("$(format-json --no-cast debug.msg=$(format-json --no-cast --scope
all-nv-pairs) log.body=${MESSAGE})" value(".tap.event"));
  set(json('{"note":"message object too long for tapping"}')
value(".tap.event") condition("$(length ${.tap.event})" > 131072));

  set("$(format-json --no-cast tap=$(format-json --no-cast tapId=''
serviceType='`service_type`' eventId=${UNIQID} labels=$(format-json --no-
cast) hostCandidate=$(format-json --no-cast) facility=${FACILITY}
format=${MSGFORMAT} hostFrom=${HOST_FROM} message=$(echo ${MESSAGE})
rawMessage=${RAWMSG} severity=${SEVERITY} source=${SOURCE} tags=${TAGS}
timestamp=${ISODATE} transport=${TRANSPORT}) event=json(${.tap.event}))"
value(".tap.formatted"));
};


# input side tapping
destination "d_axotap_input_static" {
  channel {
```

```
    filter {
      rate-limit(template("$SOURCE:$PROGRAM:$HOST:$SOURCEIP:$DESTIP")
rate(1));
    };

    rewrite {
      format_axotap_sock_message(service_type("input"));
    };

    destination {
      unix-dgram("`AXOTAP_SOCK`" persist-name("axotap_input_static")
template("${.tap.formatted}"));
    };
  };
};


log {
  destination("d_axotap_input_static");
  flags(catchall);
};


# output side tapping
destination "d_axotap_output" {
  channel {
    filter {
      rate-limit(template("$SOURCE:$PROGRAM:$HOST:$SOURCEIP:$DESTIP")
rate(1));
    };

    rewrite {
      format_axotap_sock_message(service_type("output"));
    };

    destination {
      unix-dgram("`AXOTAP_SOCK`" template("${.tap.formatted}"));
    };
  };
};
```

2. Include it in at the top of your configuration file:

```
@version: current
@include "axoflow-instrumentation.conf"
```

3. Edit every destination statement to include a `parser { metrics-output(destination(my-destination)); };` line (making sure to include the `channel` construct if your destination block does not already contain it):

```
destination my-destination {
    channel {
        parser { metrics-output(destination(my-destination)); };
        destination { file("/dev/null"); };
        };
    };
```

4. Reload the configuration of syslog-ng.

```
systemctl reload syslog-ng
```

5. To enable log tapping for the output logs (traffic that the host sends to its destinations), add the `d_axotap_output` destination in your log paths. This allows the Axolet agent to collect rate-limited log samples from that specific point in the pipeline.

   Tapping the input logs (traffic arriving to the sources of the host) needs only the configuration snippet.

> **Note**
>
> **There are multiple ways to add tapping to your configuration, which in some cases can introduce unwanted side effects. Contact us to help you create a safe tapping configuration for your use case!**

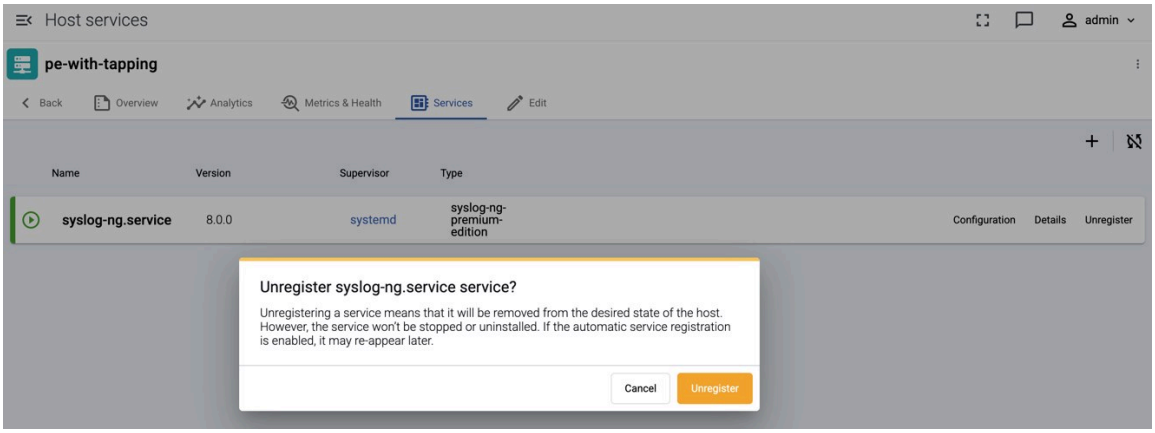
   1. Find a log path that you want to tap into and add the tap destination in a safe way, for example, by adding it in a separate inner log path:

```
log {
    source(s_udp514);
    log { destination(d_udp514); };
    log { destination(d_axotap); };
};
```

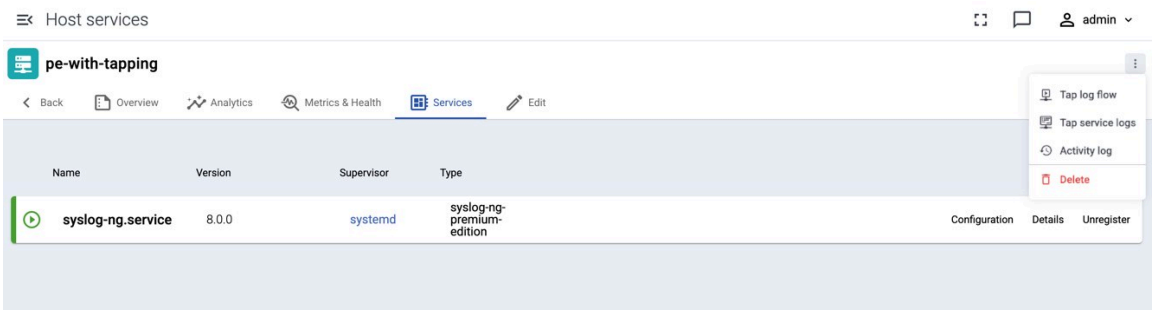
2. Reload the configuration of syslog-ng.

```
systemctl reload syslog-ng
```

3. Unregister the service so that auto service registration can discover the new tapping point:



4. Test log tapping from the top right menu:



6. If syslog-ng is sending data to an AxoRouter, create a placeholder data forwarding rule where the **Edge Selector** matches syslog-ng and the **Router connector** is the connector of the AxoRouter instance where syslog-ng is sending the data. This is needed to display syslog-ng and its data metrics on the **Topology** page, AxoConsole won't manage syslog-ng.

6.5 Axolet

Install, configure, reinstall, and remove Axolet, the agent that reports the metrics of your existing log collectors to AxoConsole.

Axolet is the agent software for Axoflow. Its primary purpose is to discover the log collector services that are running on the host (for example, AxoSyslog or SC4S), and report their operating statistics (counters) to Axoflow.

Install Axolet on hosts that already run a supported log collector, so that AxoConsole shows them on the topology with their metrics. Every AxoRouter and Axoflow agent deployment includes Axolet, so you don't install it separately on those hosts.

- To install Axolet, see Install Axolet. To set proxy and other installation options, see Advanced installation options.
- To run Axolet as the same non-root user as your log collector, see Run axolet as non-root.
- To reinstall Axolet on a cloned machine, or after a certificate rotation, see Reinstall Axolet.
- To remove Axolet, see Uninstall Axolet.

6.5.1 Install Axolet

Install Axolet on a host that runs AxoSyslog, syslog-ng, or SC4S, and approve its registration in AxoConsole.

Axolet is the agent software for Axoflow. Its primary purpose is to discover the log collector services that are running on the host (for example, AxoSyslog or SC4S), and report their operating statistics (counters) to Axoflow.

It is simple to install Axolet on individual hosts, or collectively via orchestration tools such as chef or puppet.

What the install script does

When you deploy Axolet, you run a command that installs the required software packages, configures them and sets up the connection with Axoflow.

The installer script installs the `axolet` packages, then executes the `configure-axolet` command with the right parameters. (If the packages are already installed, the installer will update them unless the `none` *package format* is selected when generating the provisioning command.)

The install script is designed to be run as root (sudo), but you can configure Axolet to run as a non-root user.

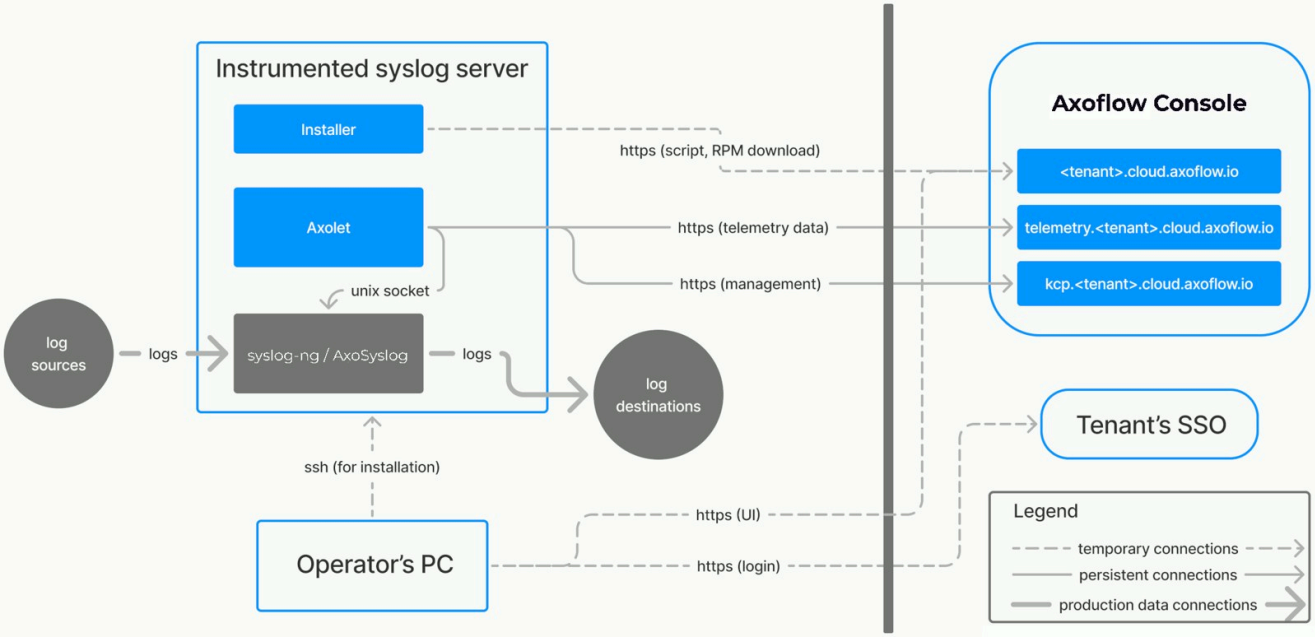
The installer script performs the following main steps:

- Executes prerequisite checks:
  - Tests the network connection with the console endpoints.
  - Checks if the operating system is supported.
- Downloads and installs the `axolet` RPM or DEB package.
  - The package contains the `axolet` and `configure-axolet` commands, and the `axolet.service` systemd unit file.
- The `configure-axolet` command is executed with a configuration snippet on its standard input which contains a token required for registering into the management platform. The command:
  - Writes the initial `/etc/axolet/config.json` configuration file.

    Note: if the file already exists it will only be overwritten if the *Overwrite config* option is enabled when generating the provisioning command.
  - Enables and starts the `axolet` service.

`axolet` performs the following main steps on its first execution:

- Generates and persists a unique identifier (GUID).
- Initiates a cryptographic handshake process to AxoConsole.
- AxoConsole issues a client certificate to Axolet, which will be stored in the above mentioned `config.json` file.
- The service waits for an approval on AxoConsole. Once you approve the host registration request, axolet starts to send telemetry data to AxoConsole. It keeps doing so as long as the agent is registered.



Prerequisites

- Supported Linux platforms for Axolet:
  - Ubuntu 22.04, 24.04, and 26.04 on x86_64 and arm64
  - Red Hat Enterprise Linux 8, 9, and 10 on x86_64 and arm64
  - Debian 12 and 13 on x86_64 and arm64

  Other compatible and derivative distributions (for example, AlmaLinux or Rocky Linux) should work as well, but we don't test them explicitly.

Network access

The host must be able to access the following domains related to the AxoConsole:

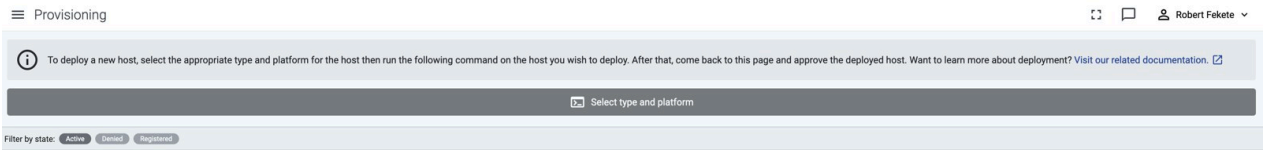
- When using AxoConsole SaaS:
  - `<your-tenant-id>.cloud.axoflow.io`: HTTPS traffic on TCP port 443, needed to download the binaries for Axoflow software (like Axolet and AxoRouter).

  - `kcp.<your-tenant-id>.cloud.axoflow.io` : HTTPS (mutual TLS) traffic on TCP port 443 for management traffic.
  - `telemetry.<your-tenant-id>.cloud.axoflow.io` : HTTPS (mutual TLS) traffic on TCP port 443, where Axolet sends the metrics of the host.
  - `us-docker.pkg.dev` : HTTPS traffic on TCP port 443, but only if you're forcing the installation script to pull the container images from the public repository. The Axolet and AxoRouter for Linux installation scripts download the images directly from AxoConsole.
- When using an on-premise AxoConsole:
  - The following domains should point to AxoConsole IP address to access Axoflow from your desktop and AxoRouter hosts:
    - `your-host.your-domain` : The main domain of your AxoConsole deployment.
    - `idp.your-host.your-domain` : A subdomain for the identity provider.
    - `kcp.your-host.your-domain` : A subdomain for management traffic.
    - `telemetry.your-host.your-domain` : A subdomain for agent telemetry.
  - The AxoConsole host must have the following **Open Ports**:
    - 80 TCP: Traefik ingress (HTTP, redirected to HTTPS).
    - 443 TCP: HTTPS for the AxoConsole UI, API, and all web traffic.
- When installing Axoflow agent for Windows or Linux:
  - `github.com` : HTTPS traffic on TCP port 443, for downloading installer packages.

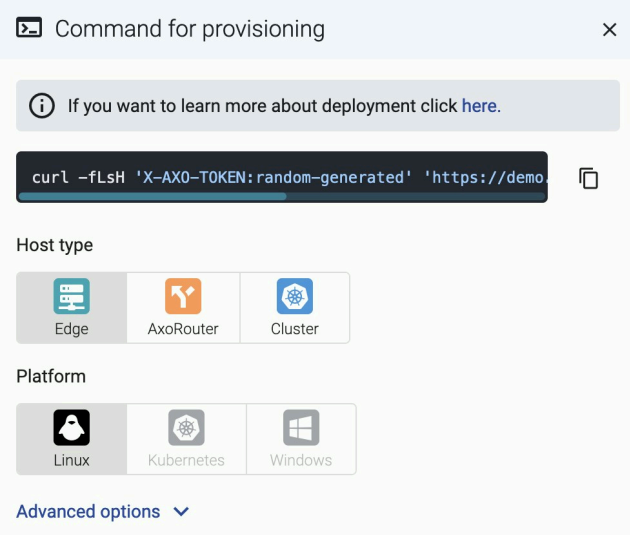
Install Axolet

To install Axolet on a host and onboard it to Axoflow, complete the following steps. If you need to reinstall Axolet for some reason, see Reinstall Axolet.

1. Open the AxoConsole at `https://<your-tenant-id>.cloud.axoflow.io/` .
2. Select **Provisioning > Select type and platform**.



3. Select **Edge > Linux**.



The `curl` command can be run manually or inserted into a template in any common software deployment package. When run, a script is downloaded that sets up the `Axolet` process to run automatically at boot time via `systemd`. For advanced installation options, see Advanced installation options.

4. Copy the deployment one-liner and run it on the host you are onboarding into Axoflow.

> ## Note
>
> **Running the provisioning command with `sudo` would mask environment variables of the calling shell. Either start the whole procedure from a root shell, or let the install script call sudo when it needs to. In other words: don't add the `sudo` command to the provisioning command.**

Example output:

```
Do you want to install Axolet now? [Y]
y
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 31.6M  100 31.6M    0     0  2127k      0  0:00:15  0:00:15 --:--:-- 2075k
Verifying packages...
Preparing packages...
axolet-0.40.0-1.aarch64
Created symlink /etc/systemd/system/multi-user.target.wants/axolet.service →
/usr/lib/systemd/system/axolet.service.
Now continue with onboarding the host on the Axoflow web UI.
```

5. On the AxoConsole, reload the **Provisioning** page. A registration request for the new host should be displayed. Accept it.
6. Axolet starts sending metrics from the host. Check the **Topology** page to see the new host.
7. Continue to onboard the host as described for your specific log collector agent.

Manage Axolet

This section describes how to start, stop and check the status of the Axolet service on Linux.

Start Axolet

To start Axolet, execute the following command. For example:

```
systemctl start axolet
```

If the service starts successfully, no output will be displayed.

The following message indicates that Axolet cannot start (see Check Axolet status):

```
Job for axolet.service failed because the control process exited with error code.
See `systemctl status axolet.service` and `journalctl -xe` for details.
```

Stop Axolet

To stop Axolet

1. Execute the following command.

   ```
   systemctl stop axolet
   ```

2. Check the status of the Axolet service (see Check Axolet status).

Restart Axolet

To restart Axolet, execute the following command.

```
systemctl restart axolet
```

Reload the configuration without restarting Axolet

To reload the configuration file without restarting Axolet, execute the following command.

```
systemctl reload axolet
```

Check the status of Axolet service

To check the status of Axolet service

1. Execute the following command.

   ```
   systemctl --no-pager status axolet
   ```

2. Check the `Active:` field, which shows the status of the Axolet service. The following statuses are possible:

   - `active (running)` - `axolet` service is up and running
   - `inactive (dead)` - `axolet` service is stopped

For AxoRouter hosts, you can check the Axolet logs from the AxoConsole using Log tapping for **Agent logs**.

Upgrade Axolet

AxoConsole raises an alert for the host when a new Axolet version is available. To upgrade to the new version, re-run the one-liner installation command you used to install Axolet, or select **Provisioning > Select type and platform** to create a new one.

Note that it can take a few minutes until the version numbers of the different services get updated on AxoConsole.

Uninstall Axolet

To remove Axolet from the host, see Uninstall Axolet.

Run axolet as non-root

You can run Axolet as non-root user, but operators must have access to the following commands:

- `/usr/bin/systemctl * axolet.service` : Controls the `axolet.service` systemd unit. Usually `*` is `start` , `stop` , `restart` , `enable` , and `status` . Used by the operators for troubleshooting.
- `/usr/local/bin/configure-axolet` : Creates initial `axolet` configuration and enables/starts the `axolet` service. Executed by the bootstrap script.
- Command to install and upgrade the `axolet` package. Executed by the bootstrap script if the packages aren't already installed.
  - On RPM-based Linux distributions: `/usr/bin/rpm -Uv axo*.rpm`
  - On DEB-based Linux distributions: `/usr/bin/dpkg -i axo*.deb`

You can permit the `syslogng` user to run these commands by running on of the following:

RPM

```
sudo tee /etc/sudoers.d/axolet <<A
syslogng ALL=(ALL) NOPASSWD: /usr/local/bin/configure-axolet
syslogng ALL=(ALL) NOPASSWD: /bin/systemctl * axolet.service
# for rpm installation:
syslogng ALL=(ALL) NOPASSWD: /usr/bin/rpm -Uv axo*.rpm
A
```

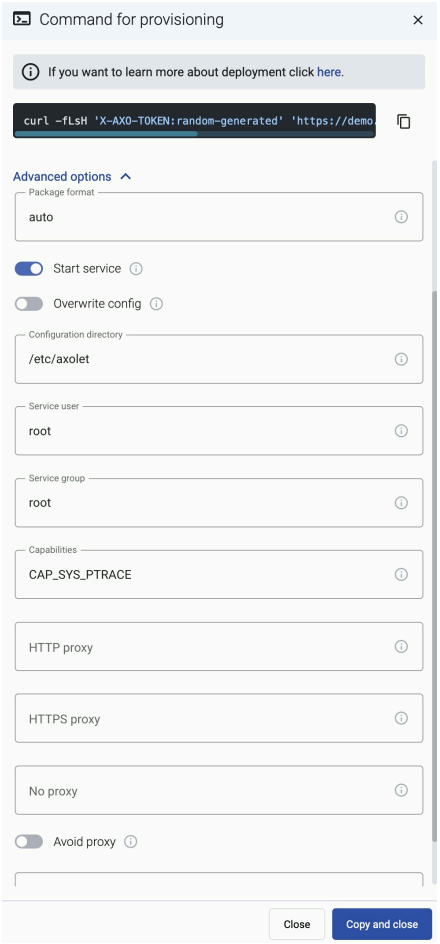
DEB

```
sudo tee /etc/sudoers.d/axolet <<A
syslogng ALL=(ALL) NOPASSWD: /usr/local/bin/configure-axolet
syslogng ALL=(ALL) NOPASSWD: /bin/systemctl * axolet.service
# for deb installation:
syslogng ALL=(ALL) NOPASSWD: /usr/bin/dpkg -i axo*.deb
A
```

6.5.2 Advanced installation options

Proxy settings and installation options for Axolet, set in AxoConsole or as environment variables before you run the installer.

When installing Axolet, you can set a number of advanced options if needed for your environment. Setting the advanced options in the AxoConsole automatically updates the one-liner command that you can copy and run.



Alternatively, before running the one-liner you can use one of the following methods:

- Set the related environment variable for the option. For example:

```
export AXOLET_USER=syslogng
export AXOLET_GROUP=syslogng
```

- Set the related URL parameter for the option. For example:

```
curl -fLsH 'X-AXO-TOKEN:random-generated' 'https://<your-tenant-
id>.cloud.axoflow.io/setup.sh?
type=AXOEDGE&platform=LINUX&user=syslogng&group=syslogng' | sh
```

Proxy settings

Use the **HTTP proxy**, **HTTPS proxy**, **No proxy** parameters to configure HTTP proxy settings for the installer. To avoid using the proxy for the Axolet service, enable the **Avoid proxy** parameter as well. Lowercase variable names are preferred because they work universally.

Installation options

You can pass the following parameters to the installation script as environment variables, or as URL parameters.

> **Note**
>
> **Running the provisioning command with `sudo` would mask environment variables of the calling shell. Either start the whole procedure from a root shell, or let the install script call sudo when it needs to. In other words: don't add the `sudo` command to the provisioning command.**

API server host

Default value:	
Environment variable	
URL parameter	`api_server_host`

Description: Override the host part of the API endpoint for the host.

Avoid proxy

Default value:	`false`

Available values:	`true` , `false`
Environment variable	`AXOLET_AVOID_PROXY`
URL parameter	`avoid_proxy`

Description: If set to `true` , the value of the `*_proxy` variables will only be used for downloading the installer, but not for the `axolet` service itself. If set to `false` , the Axolet service will use the variables from the installer.

Capabilities

Default value:	`CAP_SYS_PTRACE`
Available values:	Whitespace-separated list of capability names with `CAP_` prefix.
Environment variable	`AXOLET_CAPS`
URL parameter	`caps`

Description: Ambient Linux capabilities the `axolet` service will use.

Configuration directory

Default value:	`/etc/axolet`
Environment variable	`AXOLET_CONFIG_DIR`
URL parameter	`config_dir`

Description: The directory where the configuration files are stored.

HTTP proxy

Default value:	empty string
Environment variable	`AXOLET_HTTP_PROXY`
URL parameter	`http_proxy`

Description: Use a proxy to access AxoConsole from the host.

HTTPS proxy

Default value:	empty string
Environment variable	`AXOLET_HTTPS_PROXY`
URL parameter	`https_proxy`

Description: Use a proxy to access AxoConsole from the host.

Initial labels

Default value:	empty string
Environment variable	`AXOLET_INITIAL_LABELS`
URL parameter	`initial_labels`

Description: Comma-separated list of `key=value` labels. These labels will be suggested for the host when you add the source to AxoConsole. For example, `product=windows,team=windows,vendor=microsoft`

No proxy

Default value:	empty string
Environment variable	`AXOLET_NO_PROXY`
URL parameter	`no_proxy`

Description: Destinations that should be reached directly, without going through the proxy.

Overwrite config

Default value:	`false`
Available values:	`true` , `false`
Environment variable	`AXOLET_CONFIG_OVERWRITE`

URL parameter	`config_overwrite`

Description: If set to `true`, the configuration process will overwrite existing configuration (`/etc/axolet/config.json`). This means that the agent will get a new GUID and it will require approval on the AxoConsole.

Package format

Default value:	`auto`
Available values:	`auto`, `dep`, `rpm`, `tar`, `none`
Environment variable	`AXOLET_INSTALL_PACKAGE`
URL parameter	`install_package`

Description: File format of the installer package.

Service group

Default value:	`root`
Environment variable	`AXOLET_GROUP`
URL parameter	`group`

Description: Name of the group and Axolet will be running as. It should be either `root` or the group `syslog-ng` is running as.

Service user

Default value:	`root`
Environment variable	`AXOLET_USER`
URL parameter	`user`

Description: Name of the user Axolet will be running as. It should be either `root` or the user `syslog-ng` is running as. See also Run axolet as non-root.

Start service

Default value:	`AXOLET_START=true`
Available values:	`true` , `false`
Environment variable	`AXOLET_START`
URL parameter	`start`

Description: Start axolet agent at the end of installation. Use false for preparing golden images. In this case `axolet` will generate a new GUID on the first boot after cloning the image.

If you are preparing a host for cloning with Axolet already installed, set the following environment variable in your (root) shell session, before running the one-liner command. For example:

```
export START_AXOLET=false
curl ... # Run the command copied from the Provisioning page
```

This way Axolet will only start and initialize after the first reboot.

6.5.3 Run axolet as non-root

Run Axolet as the same non-root user as your log collector, and grant operators the commands they need.

If the log collector agent (AxoSyslog or syslog-ng) is running as a non-root user, you may want to configure the Axolet agent to run as the same user. To do that, set the `AXOLET_USER` and `AXOLET_GROUP` environment variables to the user's username and groupname. For details, see Advanced installation options.

Operators will need to have access to the following commands:

- `/usr/bin/systemctl * axolet.service` : Controls the `axolet.service` systemd unit. Usually `*` is `start` , `stop` , `restart` , `enable` , and `status` . Used by the operators for troubleshooting.
- `/usr/local/bin/configure-axolet` : Creates initial `axolet` configuration and enables/starts the `axolet` service. Executed by the bootstrap script.
- Command to install and upgrade the `axolet` package. Executed by the bootstrap script if the packages aren't already installed.
  - On RPM-based Linux distributions: `/usr/bin/rpm -Uv axo*.rpm`
  - On DEB-based Linux distributions: `/usr/bin/dpkg -i axo*.deb`

For example, you can permit the `syslogng` user to run these commands by running the following commands:

RPM

```
sudo tee /etc/sudoers.d/axolet <<A
syslogng ALL=(ALL) NOPASSWD: /usr/local/bin/configure-axolet
syslogng ALL=(ALL) NOPASSWD: /bin/systemctl * axolet.service
# for rpm installation:
syslogng ALL=(ALL) NOPASSWD: /usr/bin/rpm -Uv axo*.rpm
A
```

DEB

```
sudo tee /etc/sudoers.d/axolet <<A
syslogng ALL=(ALL) NOPASSWD: /usr/local/bin/configure-axolet
syslogng ALL=(ALL) NOPASSWD: /bin/systemctl * axolet.service
# for deb installation:
syslogng ALL=(ALL) NOPASSWD: /usr/bin/dpkg -i axo*.deb
A
```

6.5.4 Uninstall Axolet

Remove Axolet from a Linux host without stopping the log collector that runs on it.

To remove Axolet from a Linux host, complete the following steps. Uninstalling Axolet doesn't affect the log collector running on the host (for example, AxoSyslog, syslog-ng, or SC4S): it keeps running, but AxoConsole doesn't receive its metrics anymore.

1. Log in to the host and stop the `axolet` service:

   ```
   sudo systemctl disable --now axolet
   ```

   Removing the package stops the service as well. Stopping it first makes sure that it doesn't register the host again after you delete it from AxoConsole.

2. Delete the host from AxoConsole, so AxoConsole stops managing it.

   1. Open AxoConsole and select **Sources**. Alternatively, select ⌘/Ctrl + K, then enter the name of the host.
   2. Select the name of the host.
   3. Select ⋮ **Host Options > Delete**.
   4. Select **Delete and revoke**. Revoking the host prevents it from registering again with its current identity. If you plan to reinstall on the same host later, select **Delete** to keep the registration.

3. Remove the Axolet package:

   - On Red Hat Enterprise Linux 8 and 9, AlmaLinux, and Rocky Linux:

     ```
     sudo dnf remove axolet
     ```

   - On Red Hat Enterprise Linux 7:

     ```
     sudo yum remove axolet
     ```

   - On Debian and Ubuntu:

     ```
     sudo apt remove axolet
     ```

Then reload the systemd configuration, so systemd drops the units of the removed packages:

```
sudo systemctl daemon-reload
```

4. (Optional) Delete the configuration of Axolet. The package doesn't remove it, so you can reinstall Axolet and keep its identity. If you don't plan to reinstall it, run:

```
sudo rm -rf /etc/axolet
```

If you've configured Axolet to run as non-root, also delete the sudoers file you've created for it. Depending on which instructions you've followed, the file is called `axolet` or `configure-axoflow`:

```
sudo rm -f /etc/sudoers.d/axolet /etc/sudoers.d/configure-axoflow
```

6.5.5 Reinstall Axolet

Reinstall Axolet on a cloned machine, or recover it after the root CA certificate of AxoConsole was rotated.

Reinstall Axolet to a (cloned) machine

To re-install Axolet on a (cloned) machine that has an earlier installation running, complete the following steps.

1. Log in to the host as root and execute the following commands:

```
systemctl stop axolet
rm -r /etc/axolet/
```

2. Follow the regular installation steps described in Axolet.

Recover Axolet after the root CA certificate was rotated

1. Log in to the host as root and execute the following commands:

```
export AXOLET_KEEP_GUID=y AXOLET_CONFIG_OVERWRITE=y PATH=$PATH:/usr/local/bin
```

2. Run the one-liner from the AxoConsole **Provisioning** page in the same shell.

3. The installation may result in an error message, but Axolet should eventually recover. You can check by running:

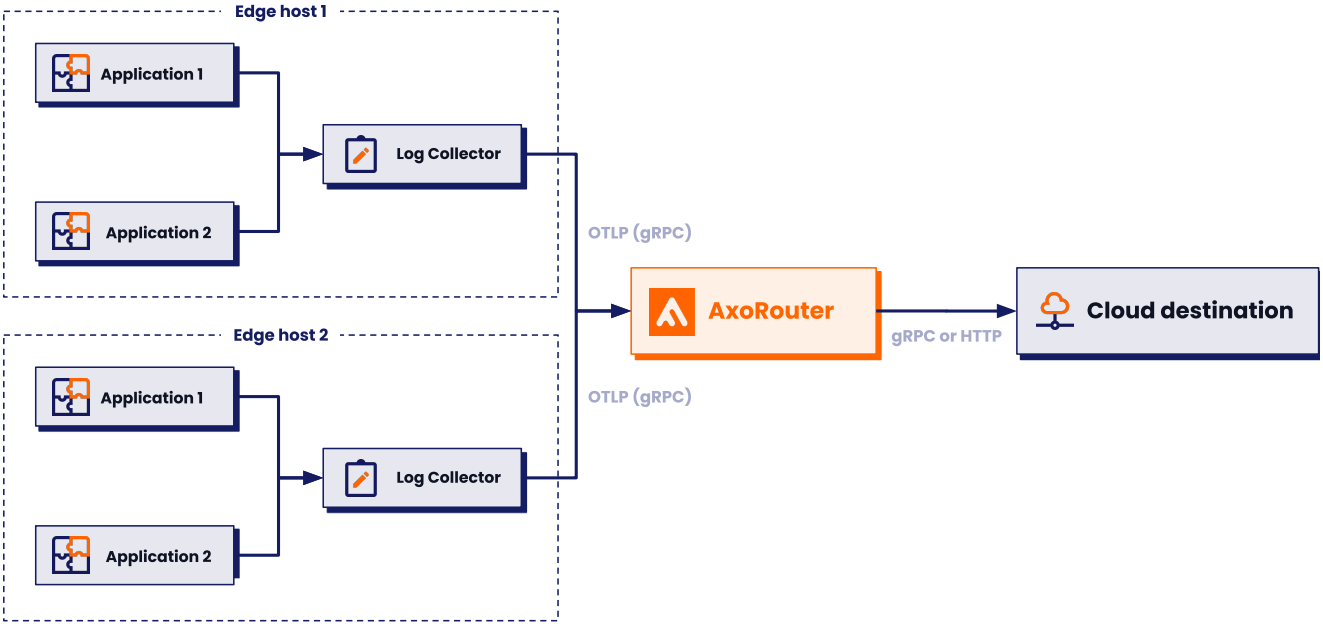
```
journalctl -b -u axolet -n 20 -f
```

6.6 Agent for Linux edge hosts

Install the Axoflow agent on Linux edge hosts to collect files and systemd journals, and check its requirements and limitations.

Axoflow provides Axoflow agent for Linux (a customized OpenTelemetry Collector distribution) to collect data from Linux-based edge hosts.

Axoflow agent for Linux can collect data from files and systemd journals.



What the installer does

When you deploy axolet, you run a command that installs the required software packages, configures them and sets up the connection with Axoflow.

The installer script performs the following main steps:

- Executes prerequisite checks:
  - Tests the network connection with the console endpoints.
  - Checks if the operating system is supported.
- Downloads the installers (`.rpm` or `.deb`) of the Axolet agent and the Axoflow agent for Linux.
- The installer script installs the packages. If the packages are already installed, the installer will update them to the latest version.

The installer installs:

- The collector agent (by default) to `/usr/bin/axoflow-otel-collector` .
- A default configuration file to `/etc/axoflow-otel-collector/config.yaml` .

- The `axolet` management agent (by default) to `/usr/local/bin/axolet`.

`axolet` performs the following main steps on its first execution:

- Generates and persists a unique identifier (GUID).
- Initiates a cryptographic handshake process to AxoConsole.
- AxoConsole issues a client certificate to axolet, which will be stored in the above mentioned `config.json` file.
- The service waits for an approval on AxoConsole. Once you approve the host registration request, axolet starts to manage the local services and send telemetry data to AxoConsole. It keeps doing so as long as the agent is registered.

> **Note**
>
> **Note that AxoRouter and Axoflow agent collects detailed, real-time metrics about the data-flows – giving you observability over the health of the security data pipeline and its components. Your security data remains in your self-managed cloud or in your on-prem instance where your sources, destinations, Axoflow agents, and AxoRouters are running, only metrics are forwarded to AxoConsole.**

To install Axoflow agent on a Linux host, complete the following steps. For other platforms, see Provision pipeline elements.

Prerequisites

- Supported Linux platforms for Axoflow agent:
  - Ubuntu 22.04, 24.04, and 26.04 on x86_64 and arm64
  - Red Hat Enterprise Linux 8, 9, and 10 on x86_64 and arm64
  - Debian 12 and 13 on x86_64 and arm64

  Other compatible and derivative distributions (for example, AlmaLinux or Rocky Linux) should work as well, but we don't test them explicitly.
- When using Axoflow agent with an on-premises AxoConsole deployment, you must prepare the Axoflow agent host

Network access

The host must be able to access the following domains related to the AxoConsole:

- When using AxoConsole SaaS:

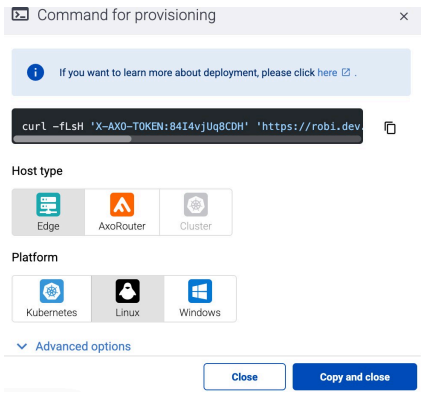
  - `<your-tenant-id>.cloud.axoflow.io` : HTTPS traffic on TCP port 443, needed to download the binaries for Axoflow software (like Axolet and AxoRouter).
  - `kcp.<your-tenant-id>.cloud.axoflow.io` : HTTPS (mutual TLS) traffic on TCP port 443 for management traffic.
  - `telemetry.<your-tenant-id>.cloud.axoflow.io` : HTTPS (mutual TLS) traffic on TCP port 443, where Axolet sends the metrics of the host.
  - `us-docker.pkg.dev` : HTTPS traffic on TCP port 443, but only if you're forcing the installation script to pull the container images from the public repository. The Axolet and AxoRouter for Linux installation scripts download the images directly from AxoConsole.
- When using an on-premise AxoConsole:
  - The following domains should point to AxoConsole IP address to access Axoflow from your desktop and AxoRouter hosts:
    - `your-host.your-domain` : The main domain of your AxoConsole deployment.
    - `idp.your-host.your-domain` : A subdomain for the identity provider.
    - `kcp.your-host.your-domain` : A subdomain for management traffic.
    - `telemetry.your-host.your-domain` : A subdomain for agent telemetry.
  - The AxoConsole host must have the following **Open Ports**:
    - 80 TCP: Traefik ingress (HTTP, redirected to HTTPS).
    - 443 TCP: HTTPS for the AxoConsole UI, API, and all web traffic.
- When installing Axoflow agent for Windows or Linux:
  - `github.com` : HTTPS traffic on TCP port 443, for downloading installer packages.
- To transport data to an AxoRouter, Axoflow agent must be able to access the OpenTelemetry Connector of the AxoRouter. By default, the connector uses the 4317 TCP port.

Limitations

Communication between the Axoflow agent hosts and AxoRouter hosts uses the OpenTelemetry Protocol (OTLP/gRPC), but currently doesn't use TLS or authentication.

Install Axoflow agent for Linux

1. Select **Provisioning > Select type and platform**.



2. Select the type (**Edge**) and platform (**Linux**). The one-liner installation command is displayed.

   If needed, set the **Advanced options** (for example, proxy settings, or **External image pull** in air-gapped environments) to modify the installation parameters. Usually, you don't have to use advanced options unless the Axoflow support team instructs you to do so.

3. Open a terminal on the host where you want to install Axoflow agent.

4. Run the one-liner, then follow the on-screen instructions.

   Example output:

```
curl -fLsH 'X-AXO-TOKEN:random-generated' 'https://<your-tenant-
id>.cloud.axoflow.io/setup.sh?type=AXOEDGE&platform=LINUX&install_agent=true' |
sh
Do you want to install Axoflow agents now? [Y]
Y

Verifying packages...
Preparing packages...
axolet-0.90.0.x86_64
The unique identifier of this installation: 9hrtcd8bz6u61aihd2zd

Verifying packages...
Preparing packages...
axoflow-otel-collector-0.129.0~axoflow.5-1.x86_64
Created symlink /etc/systemd/system/multi-user.target.wants/axoflow-otel-
collector.service → /usr/lib/systemd/system/axoflow-otel-collector.service.
Axolet service is running.
Now continue with onboarding the host on the Axoflow web UI.
```

5. Verify that the `axolet` and `axoflow-otel-collector` services are running by running `sudo systemctl list-units | grep axo`

   Example output:

   ```
   axoflow-otel-collector.service
   loaded active     running         Axoflow Otel Collector
   axolet.service
   loaded activating start     start Axoflow agent
   ```

6. On the AxoConsole, reload the **Provisioning** page. A registration request for the new host should be displayed. Accept it.

7. The host now appears on the **Topology** page. To collect data from the host, the host must match the edge selector of an Edge collection rule.

Metadata fields

The AxoRouter connector that receives data from Axoflow agent adds the following fields to the `meta` variable:

field	Value
meta.connector.type	`otlp`
meta.connector.name	`<name of the connector>`
meta.product	`opentelemetry`
meta.platform	`linux`

6.6.1 Manage the Linux agent

Start, stop, and check the services of the Axoflow agent on Linux, and upgrade or uninstall the agent.

This section describes how to start, stop and check the status of the `axolet` and `axoflow-otel-collector` services on Linux.

Start the service

To start the `axolet` or `axoflow-otel-collector` service, execute the following commands:

- `systemctl start axolet`
- `systemctl start axoflow-otel-collector`

Alternatively, you can restart the service from AxoConsole:

1. Find the host on the **Topology** or the **Sources** page. Alternatively, select `⌘/Ctrl + K` and enter the name of the host.
2. Select the name of the host.
3. Select **Services > Restart**.

Stop the service

To stop the `axolet` or `axoflow-otel-collector` service, execute the following commands:

- `systemctl stop axolet`
- `systemctl stop axoflow-otel-collector`

Check the status of the service

To check the status of the `axolet` or `axoflow-otel-collector` service, execute the following commands:

- `systemctl status axolet`
- `systemctl status axoflow-otel-collector`

For more detailed logs, run:

- `journalctl -u axolet`
- `journalctl -u axoflow-otel-collector`

Upgrade Axoflow agent

AxoConsole raises an alert for the host when a new Axoflow agent version is available. To upgrade to the new version, re-run the one-liner installation command you used to install Axoflow agent, or select **Provisioning > Select type and platform** to create a new one (make sure to enable the **Advanced options > Install Collector agent** option).

Note that it can take a few minutes until the version numbers of the different services get updated on AxoConsole.

Uninstall Axoflow agent

To remove Axoflow agent from the host, see Uninstall the Linux agent.

6.6.2 Advanced installation options

Proxy settings and installation options for the Axoflow agent for Linux, set in AxoConsole or as environment variables.

When installing Axoflow agent for Linux, you can set a number of advanced options if needed for your environment. Setting the advanced options in the AxoConsole automatically updates the one-liner command that you can copy and run.

Alternatively, before running the one-liner you can use one of the following methods:

- Set the related environment variable for the option. For example:

```
export AXOLET_INITIAL_LABELS = 'team=networking,region=us-east'
```

- Set the related URL parameter for the option. For example:

```
curl -fLsH 'X-AXO-TOKEN:random-generated' 'https://<your-tenant-
id>.cloud.axoflow.io/setup.sh?
type=AXOEDGE&platform=LINUX&initial_labels=initial_labels=team%3Dnetworking%2Cr
egion%3Dus-east' | sh
```

Proxy settings

Use the **HTTP proxy**, **HTTPS proxy**, **No proxy** parameters to configure HTTP proxy settings for the installer. To avoid using the proxy for the Axolet service, enable the **Avoid proxy** parameter as well. Lowercase variable names are preferred because they work universally.

Installation options

You can pass the following parameters to the installation script as environment variables, or as URL parameters.

API server host

Default value:

Environment variable	
URL parameter	`api_server_host`

Description: Override the host part of the API endpoint for the host.

Avoid proxy

Default value:	`false`
Available values:	`true` , `false`
Environment variable	`AXOLET_AVOID_PROXY`
URL parameter	`avoid_proxy`

Description: If set to `true` , the value of the `*_proxy` variables will only be used for downloading the installer, but not for the `axolet` service itself. If set to `false` , the Axolet service will use the variables from the installer.

Capabilities

Default value:	`CAP_SYS_PTRACE`
Available values:	Whitespace-separated list of capability names with `CAP_` prefix.
Environment variable	`AXOLET_CAPS`
URL parameter	`caps`

Description: Ambient Linux capabilities the `axolet` service will use.

Configuration directory

Default value:	`/etc/axolet`
Environment variable	`AXOLET_CONFIG_DIR`
URL parameter	`config_dir`

Description: The directory where the configuration files are stored.

HTTP proxy

Default value:	empty string
Environment variable	`AXOLET_HTTP_PROXY`
URL parameter	`http_proxy`

Description: Use a proxy to access AxoConsole from the host.

HTTPS proxy

Default value:	empty string
Environment variable	`AXOLET_HTTPS_PROXY`
URL parameter	`https_proxy`

Description: Use a proxy to access AxoConsole from the host.

Initial labels

Default value:	empty string
Environment variable	`AXOLET_INITIAL_LABELS`
URL parameter	`initial_labels`

Description: Comma-separated list of `key=value` labels. These labels will be suggested for the host when you add the source to AxoConsole. For example, `product=windows,team=windows,vendor=microsoft`

No proxy

Default value:	empty string
Environment variable	`AXOLET_NO_PROXY`
URL parameter	`no_proxy`

Description: Destinations that should be reached directly, without going through the proxy.

Overwrite config

Default value:	`false`
Available values:	`true` , `false`
Environment variable	`AXOLET_CONFIG_OVERWRITE`
URL parameter	`config_overwrite`

Description: If set to `true` , the configuration process will overwrite existing configuration (`/etc/axolet/config.json`). This means that the agent will get a new GUID and it will require approval on the AxoConsole.

Package format

Default value:	`auto`
Available values:	`auto` , `dep` , `rpm` , `tar` , `none`
Environment variable	`AXOLET_INSTALL_PACKAGE`
URL parameter	`install_package`

Description: File format of the installer package.

Service group

Default value:	`root`
Environment variable	`AXOLET_GROUP`
URL parameter	`group`

Description: Name of the group and Axolet will be running as. It should be either `root` or the group `syslog-ng` is running as.

Service user

Default value:	`root`
Environment variable	`AXOLET_USER`
URL parameter	`user`

Description: Name of the user Axolet will be running as. It should be either `root` or the user `syslog-ng` is running as. See also Run axolet as non-root.

Start service

Default value:	`AXOLET_START=true`
Available values:	`true` , `false`
Environment variable	`AXOLET_START`
URL parameter	`start`

Description: Start axolet agent at the end of installation. Use false for preparing golden images. In this case `axolet` will generate a new GUID on the first boot after cloning the image.

If you are preparing a host for cloning with Axolet already installed, set the following environment variable in your (root) shell session, before running the one-liner command. For example:

```
export START_AXOLET=false
curl ... # Run the command copied from the Provisioning page
```

This way Axolet will only start and initialize after the first reboot.

6.6.3 Configure data collection

Set up the collection and forwarding rules that tell the Axoflow agent for Linux what to collect and where to send it.

To configure the agent to collect data from the edge host and send it to an AxoRouter instance, you must have:

- a Collection rule with an Edge Selector that matches this host, and
- a data forwarding rule with an Edge Selector that matches this host.

The edge selectors of these rules are high-level policies that use dynamic host labels to determine which edge hosts match the rule.

To configure data collection on the edge host, complete the following steps.

1. Create a new Collection rule (select **Sources > Collection Rules > Add Rule**), or add the host to an existing rule.
2. Create a new Data Forwarding rule (select **Sources > Forwarding Rules > Add Rule**), or add the host to an existing rule.

6.6.4 Uninstall the Linux agent

Stop the services of the Axoflow agent, and remove it from a Linux host.

To remove Axoflow agent from a Linux host, complete the following steps.

1. Log in to the Axoflow agent host and stop the `axolet` and `axoflow-otel-collector` services:

   ```
   sudo systemctl disable --now axolet axoflow-otel-collector
   ```

   Removing the packages stops these services as well. Stopping them first makes sure that Axolet doesn't register the host again after you delete it from AxoConsole, and that nothing writes to the data directories while you remove them.

2. Delete the host from AxoConsole, so AxoConsole stops managing it.

   1. Open AxoConsole and select **Sources**. Alternatively, select ⌘/Ctrl + K, then enter the name of the host.
   2. Select the name of the host.
   3. Select ⋮ **Host Options > Delete**.
   4. Select **Delete and revoke**. Revoking the host prevents it from registering again with its current identity. If you plan to reinstall on the same host later, select **Delete** to keep the registration.

3. Remove the Axoflow agent and Axolet packages:

   - On Red Hat Enterprise Linux 8 and 9, AlmaLinux, and Rocky Linux:

     ```
     sudo dnf remove axoflow-otel-collector axolet
     ```

   - On Red Hat Enterprise Linux 7:

     ```
     sudo yum remove axoflow-otel-collector axolet
     ```

   - On Debian and Ubuntu:

     ```
     sudo apt remove axoflow-otel-collector axolet
     ```

Then reload the systemd configuration, so systemd drops the units of the removed packages:

```
sudo systemctl daemon-reload
```

4. (Optional) Delete the data, the configuration, and the system user of Axoflow agent. The packages don't remove these, so you can reinstall Axoflow agent and keep its data.

   > **CAUTION:**
   >
   > Hazard of data loss!
   >
   > `/var/lib/axoflow-otel-collector/storage` contains the disk buffer of Axoflow agent: logs that it collected but hasn't delivered to AxoRouter yet. Deleting this directory permanently deletes these logs. Make sure that you don't need this data before running the command.

   If you don't plan to reinstall it, run:

   ```
   sudo rm -rf /etc/axoflow-otel-collector /etc/axolet /var/lib/axoflow-otel-collector
   sudo userdel axoflow-otel-collector
   ```

   If you've configured Axolet to run as non-root (using the `AXOLET_USER` and `AXOLET_GROUP` advanced options), also delete the sudoers file you've created for it. Depending on which instructions you've followed, the file is called `axolet` or `configure-axoflow`:

   ```
   sudo rm -f /etc/sudoers.d/axolet /etc/sudoers.d/configure-axoflow
   ```

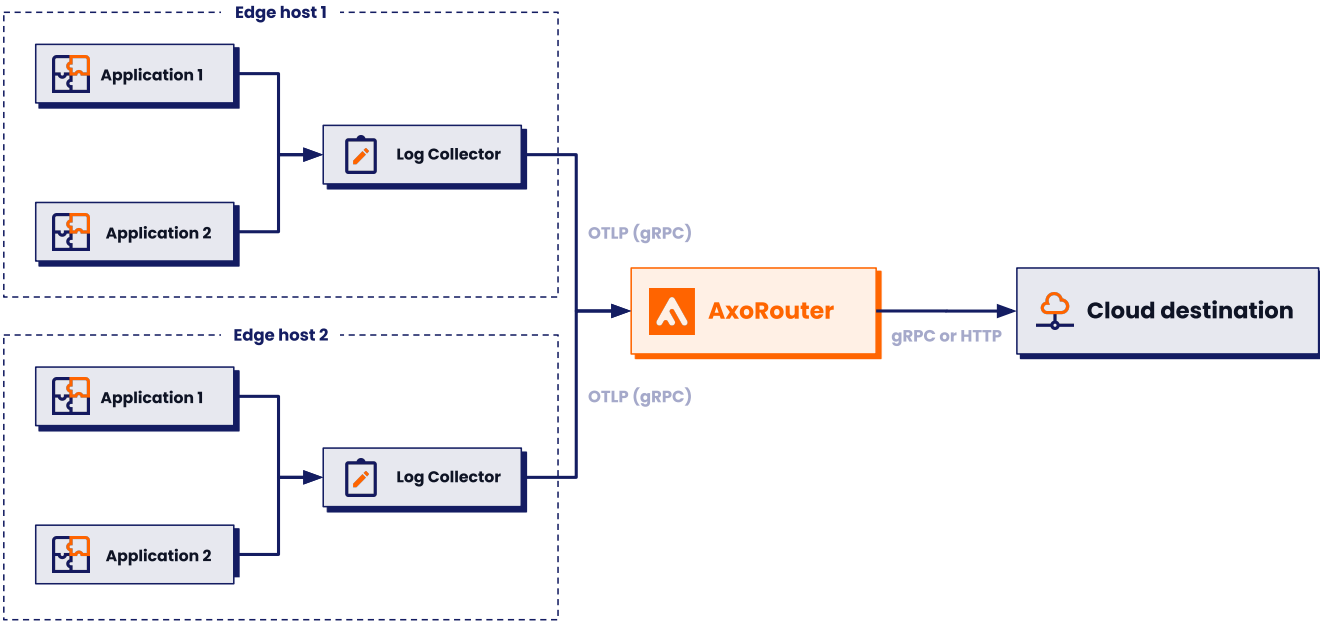
   This doesn't delete the user that Axolet ran as. Delete that user only if nothing else on the host uses it.

6.7 Windows host - agent based solution

Install the Axoflow agent on Windows hosts to collect files, Event Log, and ETW data, and check its requirements and limitations.

Axoflow provides Axoflow agent for Windows (a customized OpenTelemetry Collector distribution) to collect data from Microsoft Windows hosts.

Axoflow agent can collect data from files, Windows Event Log, and Windows Event Traces (ETW).



> **Note**
>
> **To collect data from Windows hosts without installing an agent, see Windows Event Collector (WEC)**

What the installer does

When you deploy axolet, you run a command that installs the required software packages, configures them and sets up the connection with Axoflow.

The installer script performs the following main steps:

- Executes prerequisite checks:
  - Tests the network connection with the console endpoints.
  - Checks if the operating system is supported.
- Downloads the installers (`.msi`) of the Axolet agent and the Axoflow agent for Windows.

- The installer script installs the packages. If the packages are already installed, the installer will update them to the latest version.

The installer installs:

- The collector agent (by default) to `C:\Program Files\Axoflow\OpenTelemetry Collector\axoflow-otel-collector.exe`.
- A default configuration file to `C:\ProgramData\Axoflow\OpenTelemetry Collector\config.yaml`.
- The `axolet` management agent (by default) to `C:\Program Files (x86)\Axoflow\axolet.exe`.

`axolet` performs the following main steps on its first execution:

- Generates and persists a unique identifier (GUID).
- Initiates a cryptographic handshake process to AxoConsole.
- AxoConsole issues a client certificate to axolet, which will be stored in the above mentioned `config.json` file.
- The service waits for an approval on AxoConsole. Once you approve the host registration request, axolet starts to send telemetry data to AxoConsole. It keeps doing so as long as the agent is registered.

> **Note**
>
> **Note that AxoRouter and Axoflow agent collects detailed, real-time metrics about the data-flows – giving you observability over the health of the security data pipeline and its components. Your security data remains in your self-managed cloud or in your on-prem instance where your sources, destinations, Axoflow agents, and AxoRouters are running, only metrics are forwarded to AxoConsole.**

To install Axoflow agent on a Windows host, complete the following steps. For other platforms, see Provision pipeline elements.

Prerequisites

- You'll need Administrator PowerShell access to the Windows host.
- Axoflow agent supports the following Windows versions on x86_64 architectures:
  - Windows Server 2025
  - Windows Server 2022
  - Windows Server 2019

  - Windows Server 2016
  - Windows 11
  - Windows 10
- When using Axoflow agent with an on-premises AxoConsole deployment, you must prepare the Axoflow agent host

Network access

The host must be able to access the following domains related to the AxoConsole:

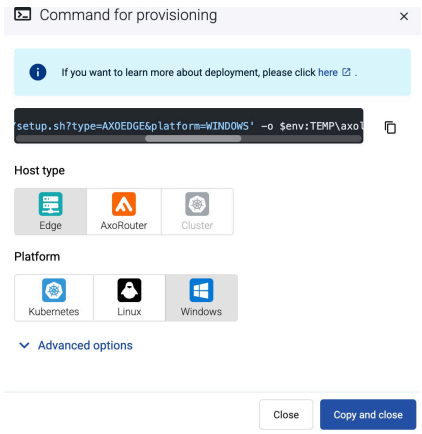
- When using AxoConsole SaaS:
  - `<your-tenant-id>.cloud.axoflow.io` : HTTPS traffic on TCP port 443, needed to download the binaries for Axoflow software (like Axolet and AxoRouter).
  - `kcp.<your-tenant-id>.cloud.axoflow.io` : HTTPS (mutual TLS) traffic on TCP port 443 for management traffic.
  - `telemetry.<your-tenant-id>.cloud.axoflow.io` : HTTPS (mutual TLS) traffic on TCP port 443, where Axolet sends the metrics of the host.
  - `us-docker.pkg.dev` : HTTPS traffic on TCP port 443, but only if you're forcing the installation script to pull the container images from the public repository. The Axolet and AxoRouter for Linux installation scripts download the images directly from AxoConsole.
- When using an on-premise AxoConsole:
  - The following domains should point to AxoConsole IP address to access Axoflow from your desktop and AxoRouter hosts:
    - `your-host.your-domain` : The main domain of your AxoConsole deployment.
    - `idp.your-host.your-domain` : A subdomain for the identity provider.
    - `kcp.your-host.your-domain` : A subdomain for management traffic.
    - `telemetry.your-host.your-domain` : A subdomain for agent telemetry.
  - The AxoConsole host must have the following **Open Ports**:
    - 80 TCP: Traefik ingress (HTTP, redirected to HTTPS).
    - 443 TCP: HTTPS for the AxoConsole UI, API, and all web traffic.
- When installing Axoflow agent for Windows or Linux:
  - `github.com` : HTTPS traffic on TCP port 443, for downloading installer packages.
- To transport data to an AxoRouter, Axoflow agent must be able to access the OpenTelemetry Connector of the AxoRouter. By default, the connector uses the 4317 TCP port.

Limitations

Communication between the Axoflow agent hosts and AxoRouter hosts uses the OpenTelemetry Protocol (OTLP/gRPC), but currently doesn't use TLS or authentication.

Install Axoflow agent for Windows

1. Select **Provisioning > Select type and platform**.



2. Select the type (**Edge**) and platform (**Windows**). The one-liner installation command is displayed.

   If needed, set the **Advanced options** (for example, proxy settings, or **External image pull** in air-gapped environments) to modify the installation parameters. Usually, you don't have to use advanced options unless the Axoflow support team instructs you to do so.

3. Open a PowerShell terminal as Administrator on the host where you want to install Axoflow agent.

4. **Only on Windows Server 2016**: Explicitly enable TLS1.2 encryption (Windows Server 2016, PowerShell 5.1 defaults to TLS 1.0 or 1.1 for web requests).

   ```
   [Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12
   ```

5. Run the one-liner, then follow the on-screen instructions.

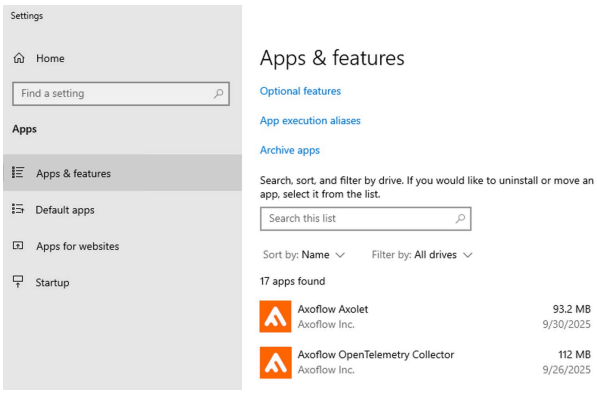
   Example output:

```
Invoke-WebRequest -Headers @{'%v' = 'X-AXO-TOKEN:xxx'} -Uri 'https://<your-tenant-id>.cloud.axoflow.io/setup.sh?type=AXOEDGE&platform=WINDOWS' -OutFile $env:TEMP\\axolet-installer.ps1; & $env:TEMP\\axolet-installer.ps1
Do you want to install Axolet now? [Y]

Installing Axoflow agent...

Axolet service is running.
Now continue with onboarding the host on the Axoflow web UI.
```

6. Verify that the `axolet` and `axoflow-otel-collector` services are running.



   If you run into errors or problems, see Troubleshooting.

7. On the AxoConsole, reload the **Provisioning** page. A registration request for the new host should be displayed. Accept it.

8. The host now appears on the **Topology** page. To collect data from the host, the host must match the edge selector of an Edge collection rule.

Metadata fields

The AxoRouter connector that receives data from Axoflow agent adds the following fields to the `meta` variable:

field	Value
meta.connector.type	`otlp`
meta.connector.name	`<name of the connector>`
meta.product	`windows`

meta.platform	`windows`

For the logs of the Windows DNS and DHCP servers, the classification of AxoRouter also sets the `service` label (`meta.service.name`) to `dns` or `dhcp` .

Troubleshooting

If you run into problems when installing Axoflow agent for Windows, complete the following steps.

1. Double-check that you're running the installation command:
    - as administrator,
    - in PowerShell, not in Cmd.
2. Set the following environment variable to store the installation logs. Adjust the directory as needed.

    ```
    $env:TEMP_DIR = "C:\Temp\axoflow-agent-install"
    ```

3. Re-run the installation command.
4. Open the Windows Event Viewer and check the **Windows Logs > System** events at the time of installation for errors or issues logged by the Service Control Manager.
5. Check that the installation logs of `axolet` and `axoflow-otel-collector` are available, for example, the `C:\Temp\axoflow-agent-install\axolet-windows-amd64.msi-install.log` and `C:\Temp\axoflow-agent-install\axoflow-otel-collector_<version-number>-axoflow.1_windows_x64.msi-install.log` files exist.
6. If you suspect that the installer is incorrectly reporting connectivity issues, disable the preflight check and re-run the installation command.

    ```
    $env:SKIP_PREFLIGHT_CHECK = 'true'
    ```

7. If needed, contact our support team.

For other problems, see Troubleshooting Axoflow.

6.7.1 Manage the Windows agent

Start, stop, and check the services of the Axoflow agent on Windows, find its logs, and upgrade or uninstall the agent.

This section describes how to start, stop and check the status of the `axolet` and `axoflow-otel-collector` services on Windows. `axolet` logs are available under `C:\Windows\Temp\axolet.log`, while Axoflow agent logs are available in the event log under `Windows Logs / Application`.

Start the service

To start the `axolet` or `axoflow-otel-collector` service, execute the following commands:

- `sc.exe start axolet`
- `sc.exe start axoflow-otel-collector`

Alternatively, you can restart the service from AxoConsole:

1. Find the host on the **Topology** or the **Sources** page. Alternatively, select `⌘/Ctrl + K` and enter the name of the host.
2. Select the name of the host.
3. Select **Services > Restart**.

Stop the service

To stop the `axolet` or `axoflow-otel-collector` service, execute the following commands:

- `sc.exe stop axolet`
- `sc.exe stop axoflow-otel-collector`

Check the status of the service

To check the status of the `axolet` or `axoflow-otel-collector` service, execute the following commands:

- `sc.exe query axolet`
- `sc.exe query axoflow-otel-collector`

Upgrade Axoflow agent

AxoConsole raises an alert for the host when a new Axoflow agent version is available. To upgrade to the new version, re-run the one-liner installation command you used to install Axoflow agent, or select **Provisioning > Select type and platform** to create a new one.

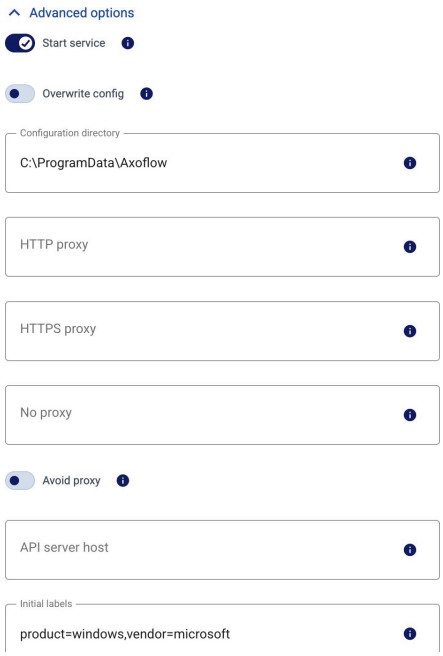
Uninstall Axoflow agent

To remove Axoflow agent from the host, see Uninstall the Windows agent.

6.7.2 Advanced installation options

Proxy settings and installation options for the Axoflow agent for Windows, set in AxoConsole or as environment variables.

When installing Axoflow agent on Windows, you can set a number of advanced options if needed for your environment. Setting the advanced options in the AxoConsole automatically updates the one-liner command that you can copy and run.



Alternatively, before running the one-liner you can use one of the following methods:

- Set the related environment variable for the option. For example:

```
$Env:AXOLET_INITIAL_LABELS = 'product=windows,team=windows,vendor=microsoft'
```

- Set the related URL parameter for the option. For example:

```
Invoke-WebRequest -Headers @{'%v' = 'X-AXO-TOKEN:xxx'} -Uri 'https://<your-tenant-id>.cloud.axoflow.io/setup.sh?type=AXOEDGE&platform=WINDOWS' -OutFile $env:TEMP\\axolet-installer.ps1; & $env:TEMP\\axolet-installer.ps1
```

Proxy settings

Use the **HTTP proxy**, **HTTPS proxy**, **No proxy** parameters to configure HTTP proxy settings for the installer. To avoid using the proxy for the Axolet service, enable the **Avoid proxy** parameter as well. Lowercase variable names are preferred because they work universally.

Installation options

You can pass the following parameters to the installation script as environment variables, or as URL parameters.

API server host

Default value:	
Environment variable	
URL parameter	`api_server_host`

Description: Override the host part of the API endpoint for the host.

Avoid proxy

Default value:	`false`
Available values:	`true` , `false`
Environment variable	`AXOLET_AVOID_PROXY`
URL parameter	`avoid_proxy`

Description: If set to `true` , the value of the `*_proxy` variables will only be used for downloading the installer, but not for the `axolet` service itself. If set to `false` , the Axolet service will use the variables from the installer.

Collector agent URL

Default value:	`https://github.com/axoflow/axoflow-otel-collector-releases/releases/download/v<version-number>/axoflow-otel-collector_<version-number>_windows_x64.msi`
Environment variable	`AXOLET_COLLECTOR_AGENT_URL`
URL parameter	`collector_agent_url`

Description: Deploy the specified agent binary.

Configuration directory

Default value:	`C:\ProgramData\Axoflow`
Environment variable	`AXOLET_CONFIG_DIR`
URL parameter	`config_dir`

Description: The directory where the configuration files are stored.

HTTP proxy

Default value:	empty string
Environment variable	`AXOLET_HTTP_PROXY`
URL parameter	`http_proxy`

Description: Use a proxy to access AxoConsole from the host.

HTTPS proxy

Default value:	empty string
Environment variable	`AXOLET_HTTPS_PROXY`
URL parameter	`https_proxy`

Description: Use a proxy to access AxoConsole from the host.

Initial labels

Default value:	`product=windows,vendor=microsoft`
Environment variable	`AXOLET_INITIAL_LABELS`
URL parameter	`initial_labels`

Description: Comma-separated list of `key=value` labels. These labels will be suggested for the host when you add the source to AxoConsole. For example, `product=windows,team=windows,vendor=microsoft`

Install collector agent

Default value:	`true`
Available values:	`true` , `false`
Environment variable	`AXOLET_INSTALL_AGENT`
URL parameter	`install_agent`

Description: Deploy the Axoflow agent. If set to `false` , only the Axolet agent will be installed.

Non-interactive installation

Default value:	`false`
Available values:	`true` , `false`
Environment variable	`AXOLET_NON_INTERACTIVE`
URL parameter	`non_interactive`

Description: Don't ask for user confirmation during the installation.

No proxy

Default value:	empty string

Environment variable	`AXOLET_NO_PROXY`
URL parameter	`no_proxy`

Description: Destinations that should be reached directly, without going through the proxy.

Overwrite config

Default value:	`false`
Available values:	`true` , `false`
Environment variable	`AXOLET_CONFIG_OVERWRITE`
URL parameter	`config_overwrite`

Description: If set to `true` , the configuration process will overwrite existing configuration (`/etc/axolet/config.json`). This means that the agent will get a new GUID and it will require approval on the AxoConsole.

Start service

Default value:	`AXOLET_START=true`
Available values:	`true` , `false`
Environment variable	`AXOLET_START`
URL parameter	`start`

Description: Start axolet agent at the end of installation. Use false for preparing golden images. In this case `axolet` will generate a new GUID on the first boot after cloning the image.

If you are preparing a host for cloning with Axolet already installed, set the following environment variable in your (root) shell session, before running the one-liner command. For example:

```
$Env:START_AXOLET = false
Invoke-WebRequest ... # Run the command copied from the Provisioning page
```

This way Axolet will only start and initialize after the first reboot.

6.7.3 Configure data collection

Set up the collection and forwarding rules that tell the Axoflow agent for Windows what to collect and where to send it.

To configure the agent to collect data from the edge host and send it to an AxoRouter instance, you must have:

- a Collection rule with an Edge Selector that matches this host, and
- a data forwarding rule with an Edge Selector that matches this host.

The edge selectors of these rules are high-level policies that use dynamic host labels to determine which edge hosts match the rule.

To configure data collection on the edge host, complete the following steps.

1. Create a new Collection rule (select **Sources > Collection Rules > Add Rule**), or add the host to an existing rule.
2. Create a new Data Forwarding rule (select **Sources > Forwarding Rules > Add Rule**), or add the host to an existing rule.

6.7.4 Uninstall the Windows agent

Stop the services of the Axoflow agent, and remove it from a Windows host.

To remove Axoflow agent from a Windows host, complete the following steps.

1. Open a PowerShell terminal as Administrator on the Axoflow agent host, and stop the `axolet` and `axoflow-otel-collector` services:

   ```
   sc.exe stop axolet
   sc.exe stop axoflow-otel-collector
   ```

   Uninstalling the packages stops these services as well. Stopping them first makes sure that Axolet doesn't register the host again after you delete it from AxoConsole, and that nothing writes to the data directories while you remove them.

2. Delete the host from AxoConsole, so AxoConsole stops managing it.

   1. Open AxoConsole and select **Sources**. Alternatively, select ⌘/Ctrl + K, then enter the name of the host.
   2. Select the name of the host.
   3. Select ⋮ **Host Options > Delete**.
   4. Select **Delete and revoke**. Revoking the host prevents it from registering again with its current identity. If you plan to reinstall on the same host later, select **Delete** to keep the registration.

3. Uninstall the Axoflow agent and Axolet packages:

   ```
   Uninstall-Package -Name "Axoflow OpenTelemetry Collector" -Force
   Uninstall-Package -Name "Axoflow Axolet" -Force
   ```

   Alternatively, uninstall **Axoflow OpenTelemetry Collector** and **Axoflow Axolet** from **Settings > Apps > Installed apps** (or **Control Panel > Programs and Features** on Windows Server).

4. (Optional) Delete the data and the configuration of Axoflow agent. The packages don't remove these, so you can reinstall Axoflow agent and keep its data.

   > **CAUTION:**
   >
   > Hazard of data loss!

> `C:\ProgramData\Axoflow\OpenTelemetry Collector\storage` contains the disk buffer of Axoflow agent: logs that it collected but hasn't delivered to AxoRouter yet. Deleting this directory permanently deletes these logs. Make sure that you don't need this data before running the command.

If you don't plan to reinstall it, run:

```
Remove-Item -Recurse -Force "C:\ProgramData\Axoflow\OpenTelemetry Collector",
"C:\ProgramData\Axoflow\axolet"
Remove-Item -Force "C:\Windows\Temp\axolet.log" -ErrorAction SilentlyContinue
```

If the `C:\ProgramData\Axoflow` folder is empty afterward, you can delete it as well.

7 Sources

Configure appliances, applications, and other sources to send their data to Axoflow, and set up the syslog, OpenTelemetry, webhook, and Windows Event Collector connectors.

The following chapters show you how to configure specific appliances, applications, and other sources to send their log data to Axoflow.

- You can find the inventory of the sources registered in AxoConsole on the **Sources** page.
- To find and add a supported source, connector rule, or collection rule, you can also use the Integrations page.
- For an overview of where the different sources are sending data, see the **Topology** page.
- To analyze your data at various points in your pipeline, see the **Analytics** page.

7.1 Generic tips

Send logs from a source that has no dedicated page: the default connectors, the open ports, and the generic syslog configuration steps.

The following section gives you a generic overview on how to configure a source to send its log data to Axoflow.

- If your source has a specific entry in the Vendors section, follow that instead of the generic procedure.
- To collect data from an edge host, see Edge collection rules.
  - Sources are hosts that are sending data to a data aggregator, like AxoRouter.
  - Edges are source hosts that are running a collector agent managed by AxoConsole, or have an Axolet agent reporting metrics from the host.

> **CAUTION:**
>
> Make sure to set data forwarding on your appliances/servers as described in this guide. Different settings like alternate message formats or ports might be valid, but can result in data loss or incorrect parsing.

Default connectors

Sources can send their data to an AxoRouter connector. By default, AxoConsole has AxoRouter connector rules that create the following connectors on AxoRouter deployments.

- OpenTelemetry
- syslog-tcp-514, syslog-udp-514, syslog-tcp-601

The following connector types are available:

- AWS CloudTrail
- AWS VPC Flow Logs
- Azure Event Hubs
- Elasticsearch Bulk API
- Falcon Platform (CrowdStrike)
- Generic HTTP
- Idira (Palo Alto Networks)
- OpenTelemetry (OTLP)
- Splunk HEC Server

- Syslog
- Vulnerability Management (Tenable)
- Webhook
- Windows Events (Windows Event Collector)

Parsing and classification is enabled for the default connectors. To create other connectors, or modify the default ones see AxoRouter connector rules. (The default connector rules have “Factory default connector rule” in their description.)

Open ports

By default, AxoRouter accepts data on the following ports (unless you’ve modified the default connector rules):

- 514 UDP and TCP for RFC3164 (BSD-syslog) and RFC5424 (IETF-syslog) formatted traffic. AxoRouter automatically recognizes and handles both formats.
- 601 TCP (RFC 3195 - Reliable Delivery for syslog) for RFC5424 (IETF-syslog) and RFC3164 (BSD-syslog) formatted traffic. AxoRouter automatically recognizes and handles both formats.
- 6514 TCP for TLS-encrypted syslog traffic (RFC 5425).
- 4317 TCP for OpenTelemetry log data.
- 5085 TCP for the Windows Event Collector (WEC) service.
- 8088 TCP for the Splunk HEC receiver (optional, only if you forward Splunk HEC traffic to AxoRouter).
- 9200 TCP for the Elasticsearch Bulk API connector (optional, only if you forward data from Elastic agents to AxoRouter).

To receive data on other ports or other protocols, configure other connector rules for the AxoRouter host.

For TLS-encrypted syslog connections, create a new connector rule or edit an existing one, and configure the keys and certificates needed to encrypt the connections. For details, see Syslog.

> **Note**
>
> **When you create a connector rule on a new port, open that port in the firewall of of every corresponding AxoRouter as well.**

Prerequisites

To configure a source to send data to Axoflow, make sure that:

- You have administrative access to the device or host.
- The date, time, and time zone are correctly set on the source.
- You have an AxoRouter deployed and configured with a Syslog connector that has parsing and classification enabled (by default, every AxoRouter has such connectors). This device is going to receive the data from the source.
- You know the IP address the AxoRouter. To find it:

  1. Open the AxoConsole.
  2. Select the **Routers** or the **Topology** page.
  3. Select on AxoRouter instance that is going to receive the logs.
  4. Check the **Networks > Address** field.

Steps

1. Log in to your device. You need administrator privileges to perform the configuration.
2. If needed, enable syslog forwarding on the device.
3. Set AxoRouter as the syslog server. Typically, you can configure the following parameters:

   - **Name or IP Address of the syslog server**: Set the address of your AxoRouter.
   - **Protocol**: If possible, set TCP or TLS.

     > **Note**
     >
     > **If you're sending data over TLS, make sure to configure a TLS-enabled connector rule in Axoflow.**

   - **Syslog Format**: If possible, set RFC5424 (or equivalent), otherwise leave the default.
   - **Port**: Set a port appropriate for the protocol and syslog format you have configured.

     By default, AxoRouter accepts data on the following ports (unless you've modified the default connector rules):

- 514 UDP and TCP for RFC3164 (BSD-syslog) and RFC5424 (IETF-syslog) formatted traffic. AxoRouter automatically recognizes and handles both formats.
- 601 TCP (RFC 3195 - Reliable Delivery for syslog) for RFC5424 (IETF-syslog) and RFC3164 (BSD-syslog) formatted traffic. AxoRouter automatically recognizes and handles both formats.
- 6514 TCP for TLS-encrypted syslog traffic (RFC 5425).
- 4317 TCP for OpenTelemetry log data.
- 5085 TCP for the Windows Event Collector (WEC) service.
- 8088 TCP for the Splunk HEC receiver (optional, only if you forward Splunk HEC traffic to AxoRouter).
- 9200 TCP for the Elasticsearch Bulk API connector (optional, only if you forward data from Elastic agents to AxoRouter).

To receive data on other ports or other protocols, configure other connector rules for the AxoRouter host.

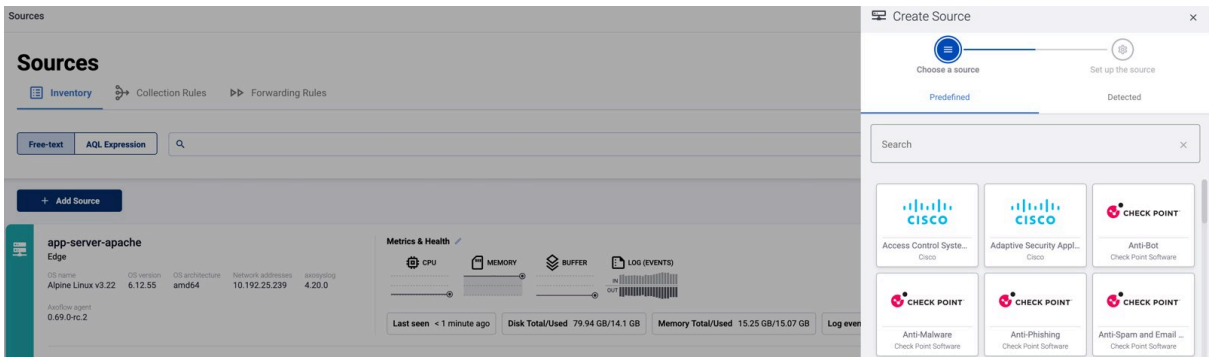
For TLS-encrypted syslog connections, create a new connector rule or edit an existing one, and configure the keys and certificates needed to encrypt the connections. For details, see Syslog.

> **Note**
>
> **When you create a connector rule on a new port, open that port in the firewall of of every corresponding AxoRouter as well.**

4. Add the source to AxoConsole.

    1. Open the AxoConsole and select **Topology**.

    2. Select **Add Item > Source**.



    - If the source is actively sending data to an AxoRouter instance, select **Detected**, then select your source(s) and select **Add Selected Source(s)**.

- Otherwise, select the vendor and product corresponding to your source from the **Predefined** sources, then enter the parameters of the source, like **IP address** and **FQDN**.



> ## Note
>
> **During log tapping, you can add hosts that are actively sending data to an AxoRouter instance by clicking Register source.**

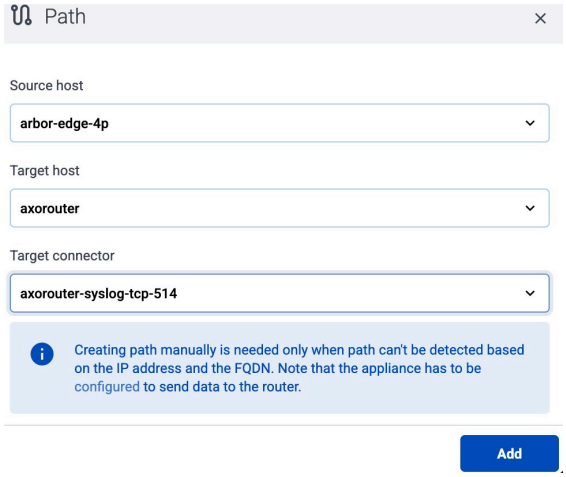
3. (Optional) Add custom labels as needed.

> **CAUTION:**
>
> AxoConsole can automatically raise an alert if a source that used to send data regularly stops sending data. To enable this alert for a source, add the `traffic_alerts:true` label to the source.

4. Select **Add**.

5. (Optional) **Add Path** manually. That's needed only when AxoConsole can't detect the path based on the IP address and the FQDN, and you haven't yet configured the source to send data to the router. If you've added the source from the **Detected** list, you can skip this step, as the path is created automatically.

> **Create a path between the source and the AxoRouter instance.**
>
> If you've added the source from the **Detected** list, you can skip this step, as the path is created automatically.
>
> 1. Select **Topology > Add Item > Path**.



2. If not set automatically, select the beginning of the path in the **Source host** field.
3. Select the target router or aggregator this source is sending its data to in the **Target host** field, for example, `axorouter`.
4. Select the **Target connector**. The connector determines how the destination receives the data (for example, using which protocol or port).
5. Select **Add**. The new path appears on the **Topology** page.



Note

If your syslog source is running `syslog-ng`, Splunk Connect for Syslog (SC4S), or AxoSyslog as its log forwarder agent, consider installing Axolet on the host and instrumenting the configuration of the log forwarder to receive detailed metrics about the host and the processed data. For details, see Manage and monitor the pipeline.

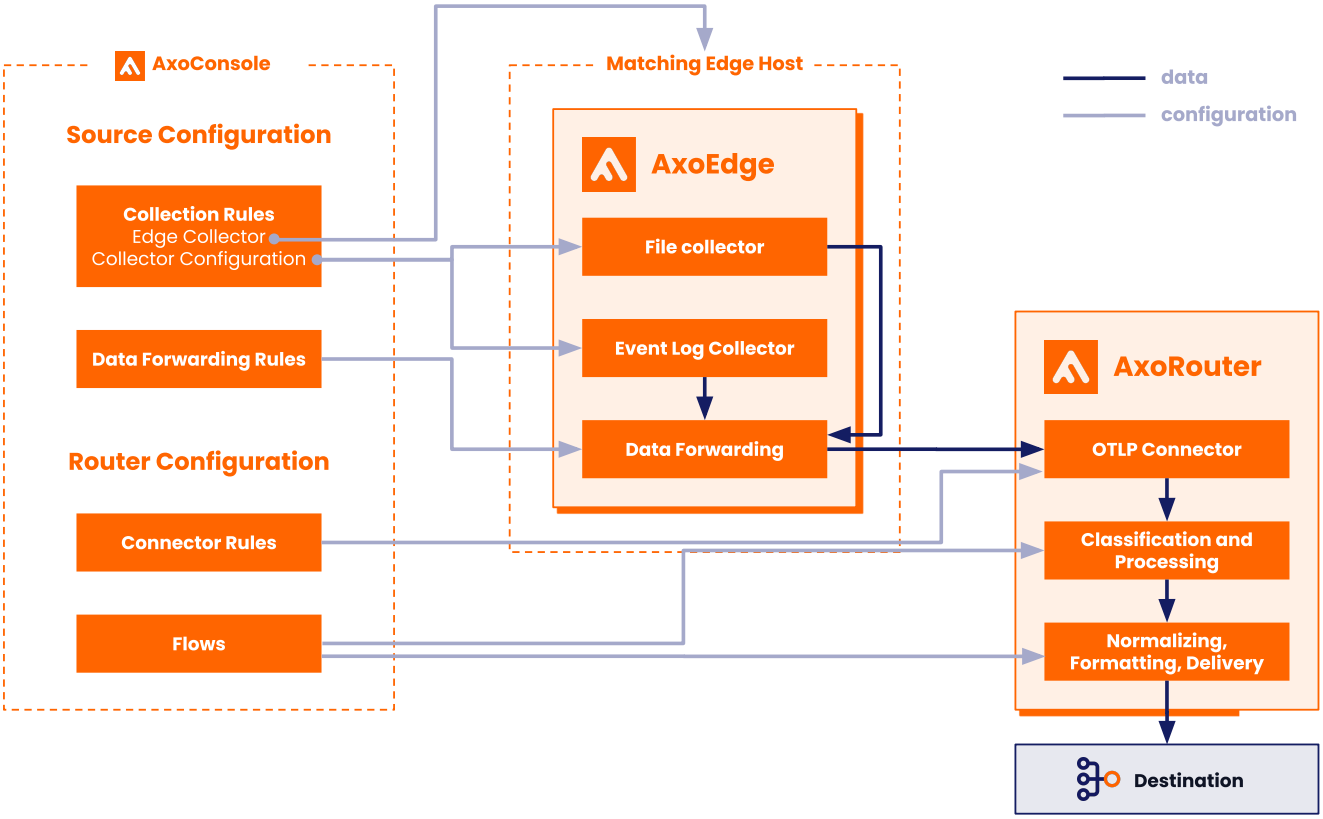
7.2 Edge collection rules

Create, modify, and extend the collection rules that tell edge hosts which files, journals, and Windows event channels to collect.

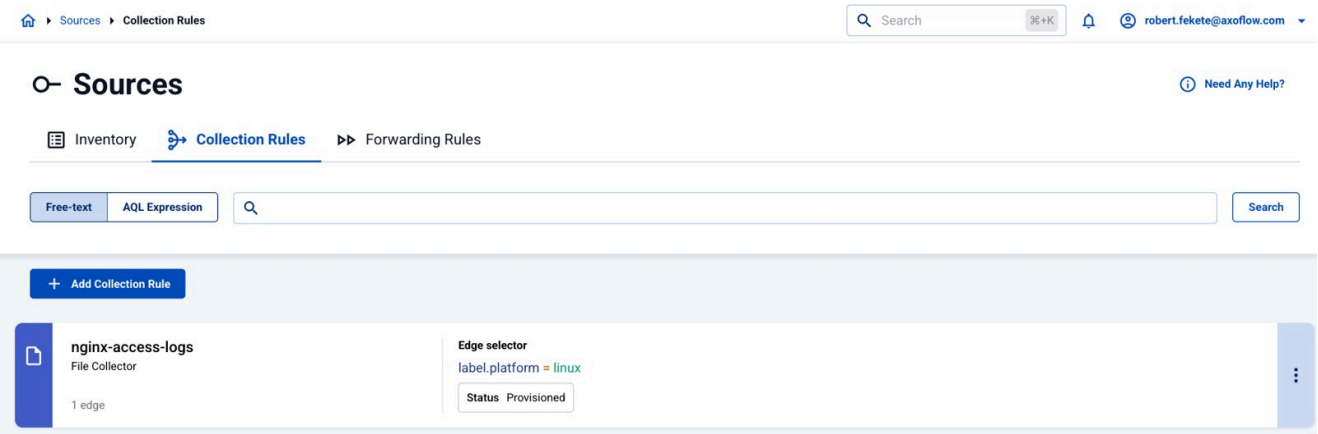
Collection rules define how edge hosts collect their local data. To collect data from non-edge sources, see AxoRouter connector rules.

- Sources are hosts that are sending data to a data aggregator, like AxoRouter.
- Edges are source hosts that are running a collector agent managed by AxoConsole, or have an Axolet agent reporting metrics from the host.

Collection rules are high-level policies that determine how data should be collected on a set of edge hosts based on dynamic host labels.



To see every collection rule configured in AxoConsole, select **Sources > Collection Rules** from the main menu.



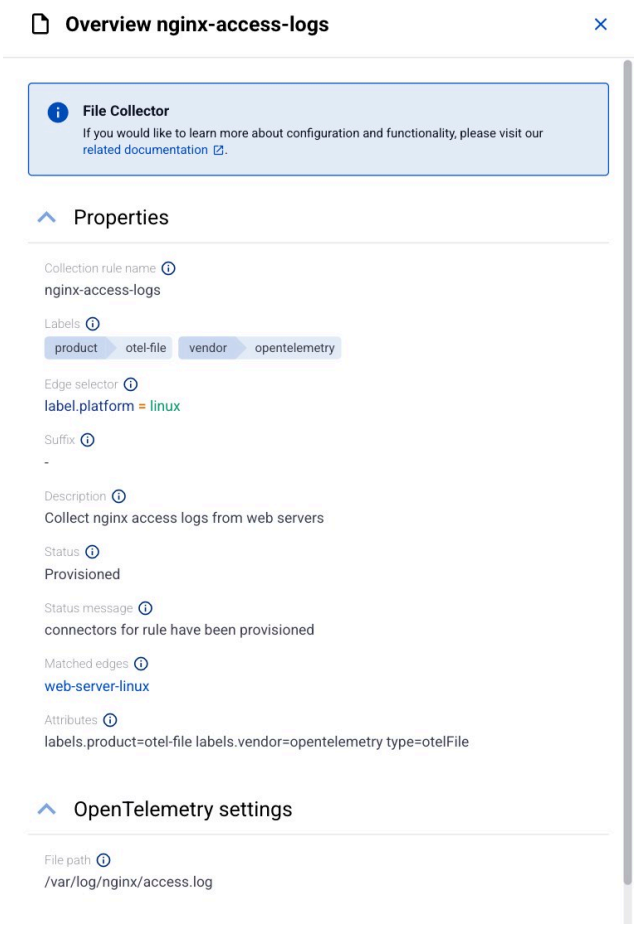
Collection rules have the following main elements:

- the way they collect data (for example, from files, or Windows Event Logs), and other specific parameters (for example, the path and filename)
- the edge selector, which determines the list of edge hosts that will create a collector based on that rule.

  You can use any labels and metadata of the edge hosts in the edge selectors, for example, the hostname, or any custom labels. For example, using the `label.product = windows` selector will create a collector only on Windows hosts.

To show the details of a collection rule, select more_vert > **Details** on its card. The details include:

- The list of **Matched edges**: the edge hosts that will have a collector based on that rule. If you select the name of a matched edge host, the **Connectors** page of that host opens, showing you the collectors configured for that host.

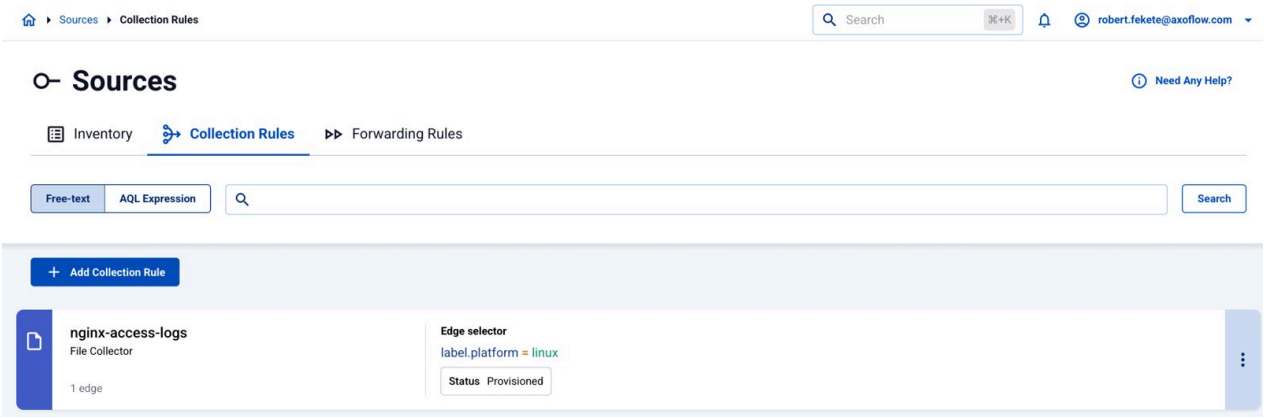


- The **Status** of the rule:
  - **Idle**: The rule doesn't match any hosts currently.
  - **Provisioned**: Connectors based on this rule were successfully provisioned for every matching host.
  - **Error**: Some error(s) occurred while provisioning connectors based on this rule. See the **Status message** field for details.
  - **Unknown**: The rule is in an unknown state.
- **Attributes**: Various significant attributes of connectors provisioned based on this rule.

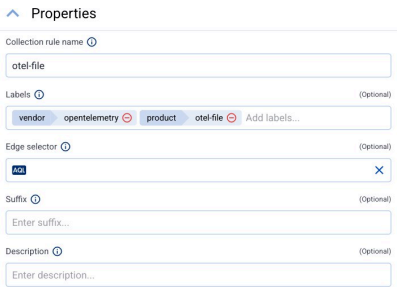
Create data collection rule

To create a new collection rule, complete the following steps.

1. Select **Sources > Collection Rules > Add Collection Rule**. (Alternatively, you can select **Add Connector > Add a data collection rule** on the **Connectors** page of an edge host.)



2. Select the type of collector you want to create. For example, **File Collector**. The following collector types are available:
   - File Collector (OpenTelemetry)
   - Journald Collector (OpenTelemetry)
   - Windows Event Log
   - Windows Event Tracing (ETW)
3. Configure the collection rule.
   1. Enter a name for the collection rule into the **Collection rule name** field.



   2. (Optional) Add labels to the collection rule.

      AxoConsole automatically adds the following selectors based on the collector type:

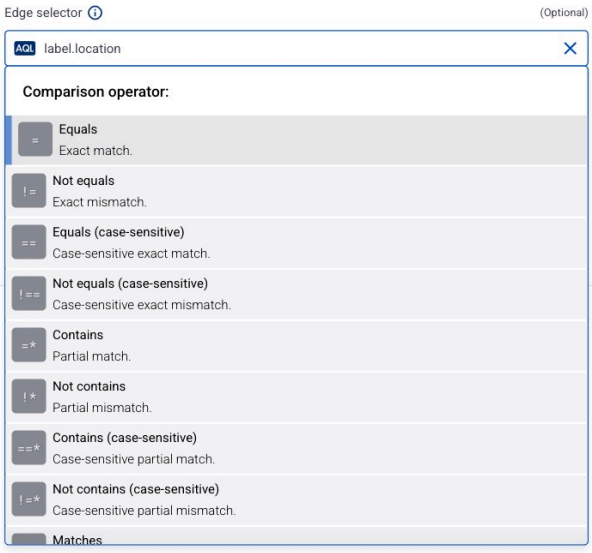
      - **File Collector**: `vendor = opentelemetry` `product = otel-file`
      - **Journald Collector**: `vendor = opentelemetry` `product = otel-journald`
      - **Windows Event Log**: `vendor = microsoft` `product = windows-event-log`
      - **Windows Event Tracing**: `vendor = microsoft` `product = windows-event-tracing`

      You can use these metrics labels as:

      - labels in **Customize Graph** on the Analytics page
      - in the **Filter labels** field during log tapping
      - in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps.

      For edge-related metrics, see the metrics beginning with `edge_connector`

3. Set the **Edge selector** for the collection rule. The selector determines which edge hosts will have a collector based on this collection rule.



   - Only edge hosts will match the rule.
   - If you leave the edge selector empty, the rule matches every edge host.
   - To select only a specific host, set the `name` field to the name of the host as selector.
   - If you set multiple fields in the selector, the collection rule will apply only to edge hosts that match all elements of the selector. (There in an AND relationship between the fields.) For example, `label.location = us-east-1 AND label.product = windows`
   - AxoConsole automatically adds the following selectors based on the collector type:
     - **Journald Collector**: `label.platform = linux`
     - **Windows Event Log**: `label.platform = windows`
     - **Windows Event Tracing**: `label.platform = windows`
4. (Optional) Enter a **Suffix** for the collection rule. AxoConsole adds the suffix to the name of the collector instances that it creates on the edge hosts. For example, if the name of a matching edge host is `edge-host-1`, and the suffix of the rule is `otel-file-collector`, the name of the collector on that host is `edge-host-1-otel-file-collector`.

   If the **Suffix** field is empty, AxoConsole uses the name of the collection rule instead.
5. (Optional) Enter a description for the rule.

4. Configure the options specific to the collector type. For details, see the specific pages:

- File Collector (OpenTelemetry)
- Journald Collector (OpenTelemetry)
- Windows Event Log
- Windows Event Tracing (ETW)

5. Select **Add**. Based on the collection rule, Axoflow automatically creates collectors on the edge hosts that match the **Edge selector**.

> **CAUTION:**
>
> Make sure to configure data forwarding rules for your edge hosts to transfer the collected data to the OpenTelemetry connector of an AxoRouter.

Modify collection rule

To modify an existing collection rule, complete the following steps.

> **Note**
>
> - You cannot directly modify the collector of an edge host, only via modifying a collection rule.
> - Modifying a collection rule affects every edge host that matches the **Edge selector** of the rule.
> - To apply an existing collection rule to a new edge host, you can:
>   - Add a label to the edge host so the **Edge selector** of the collection rule matches the new host, and/or
>   - Change the **Edge selector** of the collection rule to match the new host.

1. Find the collection rule you want to modify:
   - Select **Sources > Collection Rules** from the main menu, then select more_vert > **Edit** on the card of the collection rule you want to modify.
   - Alternatively, find the edge host whose collector you want to modify on the **Topology** page, then select **Connectors**. Find the collector you want to modify, then select more_vert > **Edit connector rule**.
2. Modify the configuration of the collection rule as needed.

> **CAUTION:**
>
> The changes are applied immediately after you click **Update**. Double-check your changes to avoid losing data.

3. Select **Update**.

Add edge host to existing collection rule

To add an edge host to an existing collection rule, you have two options, depending on the **Edge selector** of the collection rule:

- Modify the **Edge selector** field of the collection rule to include the new edge host.
- Modify the labels of the edge host so it matches the edge selector of the collection rule.

7.2.1 File Collector

Collect log files from Linux and Windows edge hosts, and look up the labels, metrics, and advanced options of the File Collector.

Collect logs from a local file that's available on the edge host.

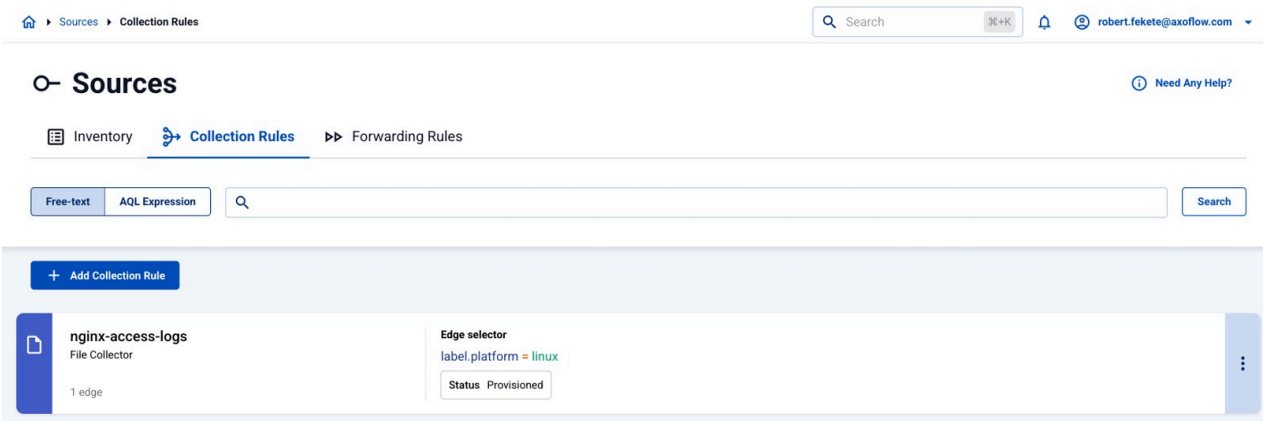
Prerequisites

This collector can be deployed to edge hosts running Axoflow agent for Linux and Axoflow agent for Windows.

Add new File Collector

To create a new collection rule that collects data from files on edge hosts, complete the following steps:

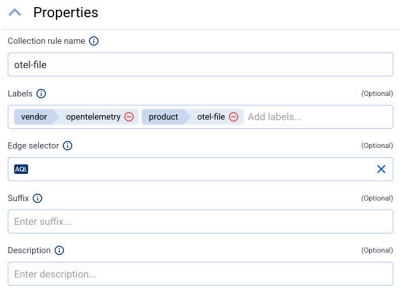
1. Select **Sources > Collection Rules > Add Collection Rule**. (Alternatively, you can select **Add Connector > Add a data collection rule** on the **Connectors** page of an edge host.)



2. Select **File Collector**.

3. Configure the collection rule.

    1. Enter a name for the collection rule into the **Collection rule name** field.



2. (Optional) Add labels to the collection rule.

   AxoConsole automatically adds the following selectors based on the collector type:

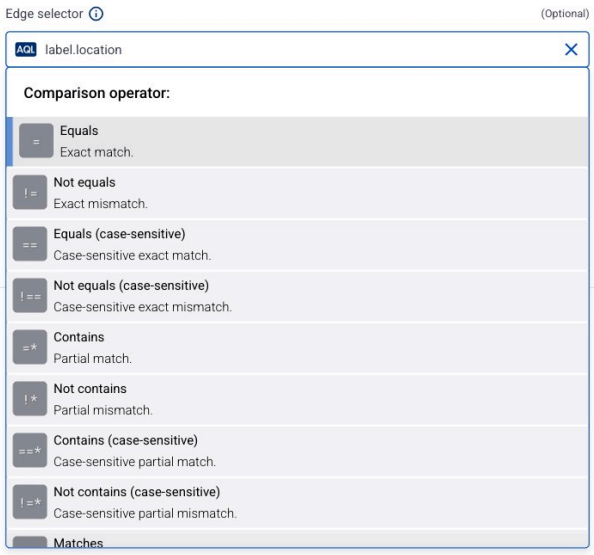
   - **File Collector**: `vendor = opentelemetry` `product = otel-file`
   - **Journald Collector**: `vendor = opentelemetry` `product = otel-journald`
   - **Windows Event Log**: `vendor = microsoft` `product = windows-event-log`
   - **Windows Event Tracing**: `vendor = microsoft` `product = windows-event-tracing`

   You can use these metrics labels as:

   - labels in **Customize Graph** on the Analytics page
   - in the **Filter labels** field during log tapping
   - in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps.

   For edge-related metrics, see the metrics beginning with `edge_connector`

3. Set the **Edge selector** for the collection rule. The selector determines which edge hosts will have a collector based on this collection rule.



   - Only edge hosts will match the rule.
   - If you leave the edge selector empty, the rule matches every edge host.
   - To select only a specific host, set the `name` field to the name of the host as selector.
   - If you set multiple fields in the selector, the collection rule will apply only to edge hosts that match all elements of the selector. (There in an AND relationship between the fields.) For example, `label.location = us-east-1 AND label.product = windows`
   - AxoConsole automatically adds the following selectors based on the collector type:

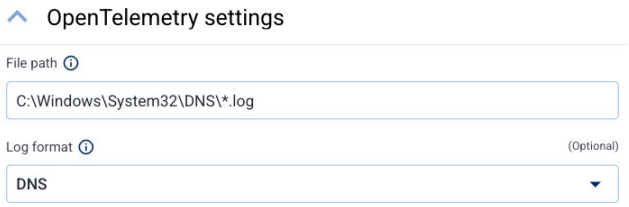
- **Journald Collector**: `label.platform = linux`
- **Windows Event Log**: `label.platform = windows`
- **Windows Event Tracing**: `label.platform = windows`

4. (Optional) Enter a **Suffix** for the collection rule. AxoConsole adds the suffix to the name of the collector instances that it creates on the edge hosts. For example, if the name of a matching edge host is `edge-host-1`, and the suffix of the rule is `otel-file-collector`, the name of the collector on that host is `edge-host-1-otel-file-collector`.

   If the **Suffix** field is empty, AxoConsole uses the name of the collection rule instead.

5. (Optional) Enter a description for the rule.

4. Enter the path of the log file, or a pattern to match multiple files into the **File path** field, for example: `C:\Windows\System32\DNS\dns.log` or `/path/to/**/*.log`



CAUTION:

On Linux hosts, the collector runs as the `axoflow-otel-collector` user, which is a member of the `adm` and `systemd-journal` groups. Make sure that the `axoflow-otel-collector` user has read access to the file you want to collect logs from. Usually, the `adm` group can read logs from the `/var/log/` directory on Debian-based systems, but not on RHEL-based systems.

You can use the following special characters:

- `*`: Matches one or more characters that aren't path separators.
- `/**/`: Matches zero or more directories.
- `?`: Matches a single non-path-separator character.
- `[class]`: Matches any single non-path-separator character from the specified class. The following classes are available:
  - `[abc123]`: Matches any single character of the specified characters.
  - `[a-z0-9]`: Matches any single alphanumeric character in the range of a-z or 0-9.

- `[^class]` or `[!class]`: Negates the class, so it matches any single character which does not match the class.

5. (Optional) If needed, set advanced options under **More options**.
6. To apply a specific parser on the messages of the log file, select it from the **Log format** field. Currently Windows DNS and DHCP log files are supported.
7. Select **Add**. Based on the collection rule, Axoflow automatically creates collectors on the edge hosts that match the **Edge selector**.

> **CAUTION:**
>
> Make sure to configure data forwarding rules for your edge hosts to transfer the collected data to the OpenTelemetry connector of an AxoRouter.

Related metrics

You can use these metrics labels as:

- labels in **Customize Graph** on the Analytics page
- in the **Filter labels** field during log tapping
- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps.

label	Value
edge_connector_name	The name of the collector that collected the message.
edge_connector_type	`otelFile`
edge_connector_label_	Labels set by the collector. By default: `vendor:opentelemety`, `product:otel-file`
edge_connector_rule_id	The ID of the Collector Rule resource in Axoflow that created the collector.
edge_flow_name	The name of the data forwarding rule that sent the message.

Advanced options

- **Exclude file pattern**: Exclude some files that match the **File path**. You can use the same special characters as in the **File path** field.

- **Exclude older than**: Exclude files whose modification time is older than the specified value, for example: `1h`, `24h`, `7d`.
- **Multi-line start pattern**: Regex pattern to identify the start of a multi-line log entry. Mutually exclusive with **Multi-line end pattern**.
- **Multi-line end pattern**: Regex pattern to identify the end of a multi-line log entry. Mutually exclusive with Multi-line start pattern.
- **Multi-line omit pattern**: If enabled, the lines matching the multiline pattern are omitted from the entry.
- **Force flush period**: Always flush the current batch if the after the specified period. Example values: `1s`, `5m`, `1h`. Default value: `500ms`
- **Encoding**: Specifies the encoding of the file being read. Default value: `utf-8`. The following values are supported:
  - `nop`: No encoding validation. Treats the file as a stream of raw bytes
  - `utf-8`: UTF-8 encoding
  - `utf-8-raw`: UTF-8 encoding without replacing invalid UTF-8 bytes
  - `utf-16le`: UTF-16 encoding with little-endian byte order
  - `utf-16be`: UTF-16 encoding with big-endian byte order
  - `ascii`: ASCII encoding
  - `big5`: The Big5 Chinese character encoding
- **Poll interval**: The duration between filesystem polls, for example: `1s`, `5m`, `1h`. Default value: `200ms`
- **Retry on failure max elapsed time**: Maximum time (including retries) to send a log batch to a downstream consumer before discarding it, for example: `1s`, `5m`, `1h`. Retrying never stops if set to `0`. Default value `0`
- **Initial buffer size**: The initial size (in KiB) of the buffer to read file headers and logs. The buffer will grow as needed; larger values may cause unnecessary memory allocation, while smaller values may require multiple copies during growth. Default value: `16KiB`
- **Max log size**: Maximum size of a log entry in megabytes. Larger log entries will be truncated. Default value: `1MiB`
- **Max concurrent files**: Maximal number of files to read from in parallel. Default value: `1024`

- **Max batches**: Maximum number of batches to keep in memory; applicable only when more files match the **File path** than the value of **Max concurrent files**. To set this option, set **Max concurrent files** as well.
- **Compression**: Specifies the compression format of the files being read. Possible values are the empty string, `gzip`, and `auto`. Use `auto` when your **File path** matches a mix of compressed and uncompressed files.
- **Start at**: Specifies where to start reading logs on startup: `beginning` or `end` of the file. Default value: `beginning`

7.2.2 Journald Collector

Collect systemd journal entries from Linux edge hosts, and look up the labels, metrics, and advanced options of the Journald Collector.

Collect logs from the journald system service of Linux-based edge hosts.

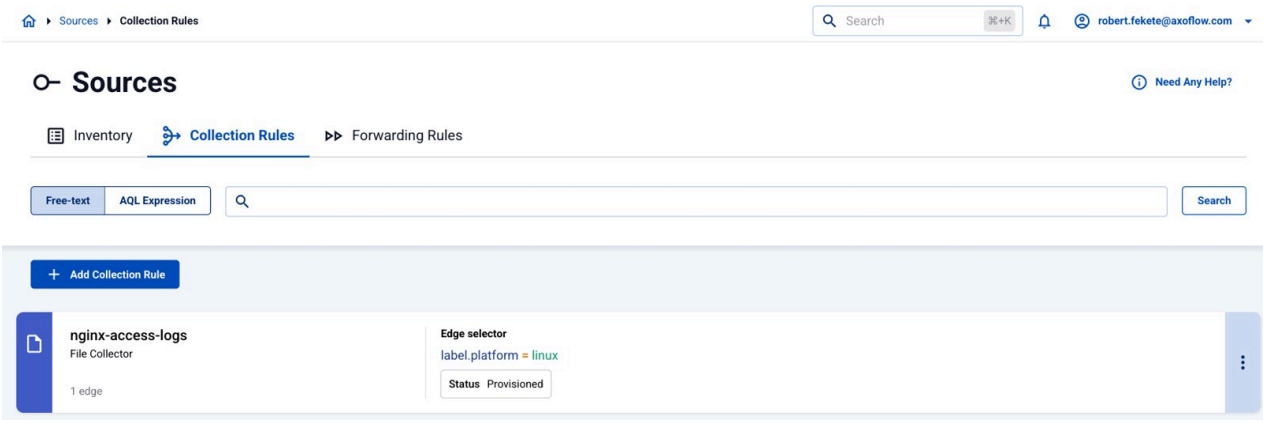
Prerequisites

This collector can be deployed to edge hosts running Axoflow agent for Linux.

Add new Journald Collector

To create a new collection rule that collects logs from journald, complete the following steps:

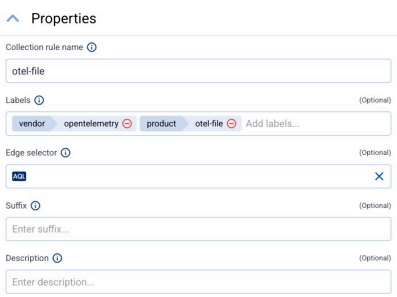
1. Select **Sources > Collection Rules > Add Collection Rule**. (Alternatively, you can select **Add Connector > Add a data collection rule** on the **Connectors** page of an edge host.)



2. Select **Journald Collector**.

3. Configure the collection rule.

    1. Enter a name for the collection rule into the **Collection rule name** field.



2. (Optional) Add labels to the collection rule.

   AxoConsole automatically adds the following selectors based on the collector type:

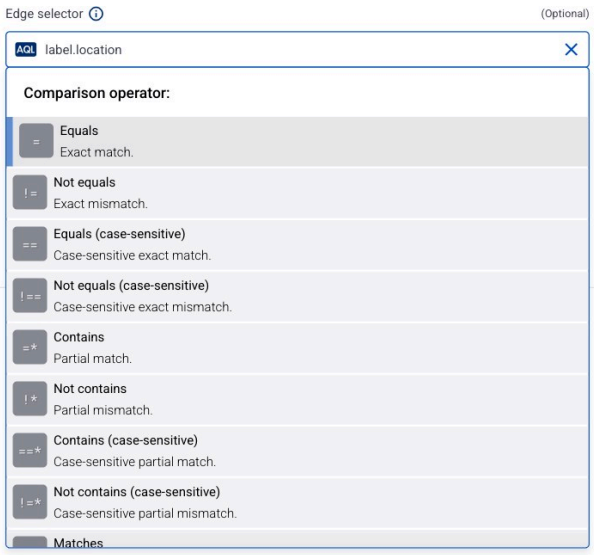
   - **File Collector**: `vendor = opentelemetry` `product = otel-file`
   - **Journald Collector**: `vendor = opentelemetry` `product = otel-journald`
   - **Windows Event Log**: `vendor = microsoft` `product = windows-event-log`
   - **Windows Event Tracing**: `vendor = microsoft` `product = windows-event-tracing`

   You can use these metrics labels as:

   - labels in **Customize Graph** on the Analytics page
   - in the **Filter labels** field during log tapping
   - in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps.

   For edge-related metrics, see the metrics beginning with `edge_connector`

3. Set the **Edge selector** for the collection rule. The selector determines which edge hosts will have a collector based on this collection rule.



   - Only edge hosts will match the rule.
   - If you leave the edge selector empty, the rule matches every edge host.
   - To select only a specific host, set the `name` field to the name of the host as selector.
   - If you set multiple fields in the selector, the collection rule will apply only to edge hosts that match all elements of the selector. (There in an AND relationship between the fields.) For example, `label.location = us-east-1 AND label.product = windows`
   - AxoConsole automatically adds the following selectors based on the collector type:

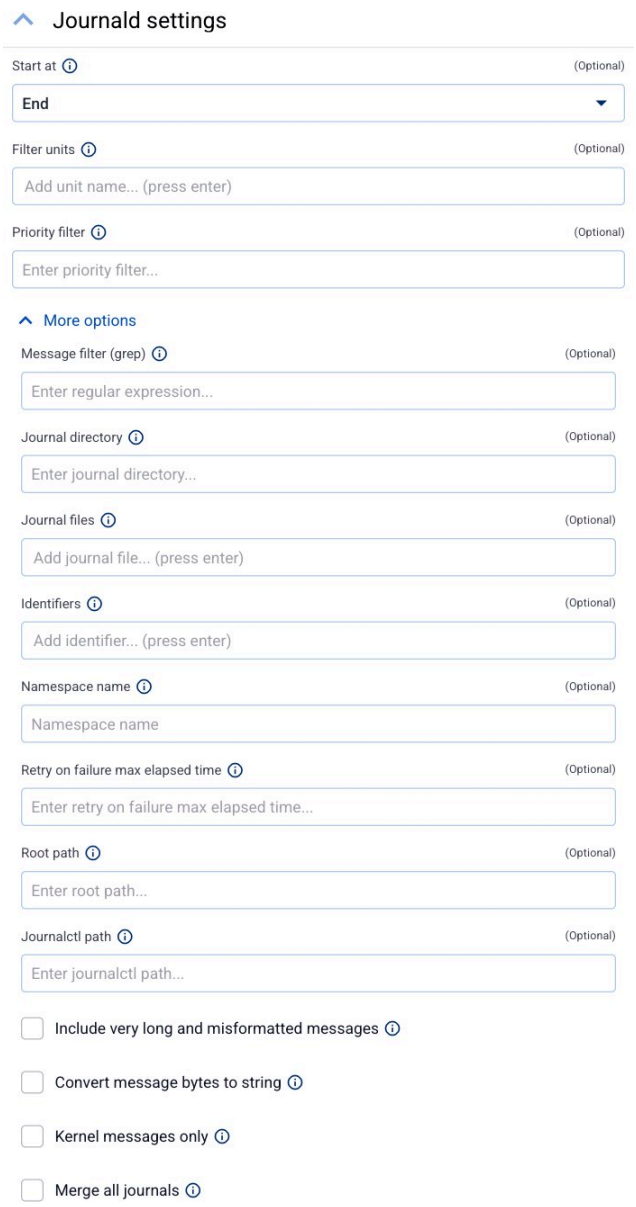
- **Journald Collector**: `label.platform = linux`
- **Windows Event Log**: `label.platform = windows`
- **Windows Event Tracing**: `label.platform = windows`

4. (Optional) Enter a **Suffix** for the collection rule. AxoConsole adds the suffix to the name of the collector instances that it creates on the edge hosts. For example, if the name of a matching edge host is `edge-host-1`, and the suffix of the rule is `otel-file-collector`, the name of the collector on that host is `edge-host-1-otel-file-collector`.

   If the **Suffix** field is empty, AxoConsole uses the name of the collection rule instead.

5. (Optional) Enter a description for the rule.

4. (Optional) To read older entries from the journal files, set **Start at** to `Beginning`. Otherwise, Axoflow agent will only forward the journal entries that are created after the collector has been deployed.



5. (Optional) To read only the entries from specific journald units, list the units in the **Filter units** field (for example, `nginx.service`). By default, Axoflow agent reads the entries of every unit. To list the units available on a host, run the following command on the host: `sudo systemctl list-units`

6. (Optional) To read only entries with the specified or higher priority, enter the priority value into the **Priority filter** field. Default value: `info` (so debug level entries are omitted). The possible values in decreasing order are: `emerg`, `alert`, `crit`, `err`, `warning`, `notice`, `info`, `debug`.

7. (Optional) If needed, set advanced options under **More options**.

8. Select **Add**. Based on the collection rule, Axoflow automatically creates collectors on the edge hosts that match the **Edge selector**.

CAUTION:

> Make sure to configure data forwarding rules for your edge hosts to transfer the collected data to the OpenTelemetry connector of an AxoRouter.

Related metrics

You can use these metrics labels as:

- labels in **Customize Graph** on the Analytics page
- in the **Filter labels** field during log tapping
- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps.

label	Value
edge_connector_name	The name of the collector that collected the message.
edge_connector_type	`otelJournald`
edge_connector_label_	Labels set by the collector. By default: `vendor:opentelemety` , `product:otel-journald`
edge_connector_rule_id	The ID of the Collector Rule resource in Axoflow that created the collector.
edge_flow_name	The name of the data forwarding rule that sent the message.

Advanced options

Note that if you set more than one filter-like fields (for example, **Priority filter** and **Identifiers**), Axoflow agent reads only entries that match all filters (there's a logical AND operator between the fields). Within a field (for example, if you specify multiple **Identifiers**) the filters have an OR relation, so any matching entry is read (unless it gets excluded by another filter).

- **Message filter (grep)**: Read only entries where the `MESSAGE` field matches the specified regular expression.
- **Journal directory**: Specifies the directory containing journal files to read entries from. Relative to the **Root path**. Default value: `/run/log/journal` or `/run/journal` , depending on the platform.
- **Journal files**: Specifies the list of journal files to read entries from. Relative to the **Root path**. By default it's empty, meaning that all files will be read.

- **Identifiers**: Read only entries of the listed message identifiers (SYSTEMD_IDENTIFIER), for example, `2`.
- **Namespace name**: Query the given namespace. See man page systemd-journald.service(8) for details.
- **Retry on failure max elapsed time**: Maximum amount of time (including retries) spent trying to send a logs batch to AxoRouter, for example, `5 minutes`. When this value is reached, the data that wasn't sent is discarded. Default value: `0` (keep retrying indefinitely)
- **Root path**: The chroot to use when executing the `journalctl` command. By default, it's empty (no chroot is used). To set a path, use an absolute path. Note that if you set a root path, other options of the collector must be set relative to the root path (for example, **Journal directory**), while others must be absolute (for example, **Journalctl path**).
- **Journalctl path**: The `journalctl` command to execute. Relative to the **Root path**, unless the **Root path** is set, in which case **Journalctl path** must be absolute. Default value: `journalctl`
- **Include very long and misformatted messages**: Read very long logs and logs with unprintable characters.
- **Convert message bytes to string**: If the `MESSAGE` field of an entry incudes an array of bytes, convert the array to string.
- **Kernel messages only**: Read only kernel messages (dmesg). This shows logs from the current boot and that match `_TRANSPORT=kernel`.
- **Merge all journals**: Read from all available journals, including remote ones.

7.2.3 Windows Event Log

Collect Windows Event Log channels from Windows edge hosts, and look up the labels, metrics, and advanced options of the collector.

Collect logs from the Event Log of the host. You can:

- read all data from specific Event Log channels,
- use a custom event filter query, or
- read data from a specific EVTX file.

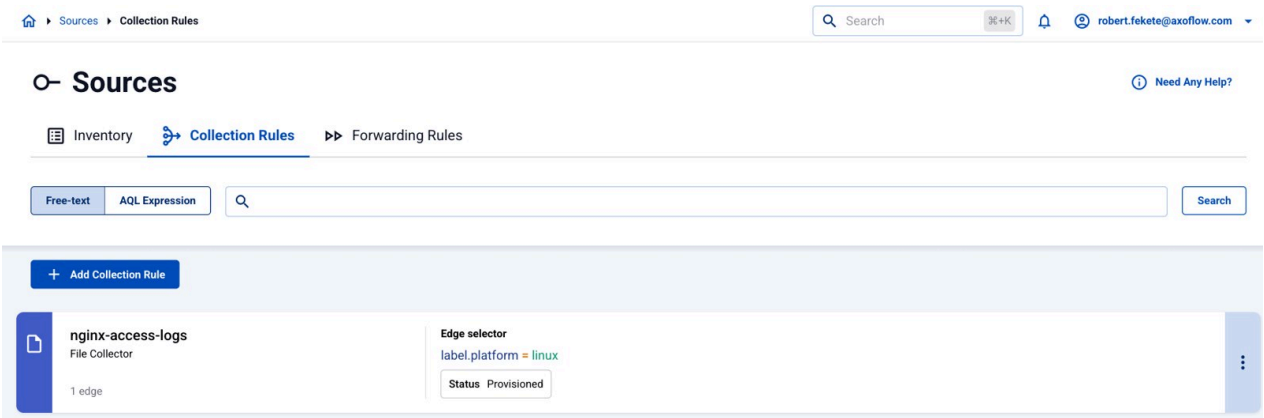
Prerequisites

This collector can be deployed to edge hosts running Axoflow agent for Windows.

Add new Event Log Collector

To create a new collection rule that collects Windows Event Logs from edge hosts, complete the following steps:

1. Select **Sources > Collection Rules > Add Collection Rule**. (Alternatively, you can select **Add Connector > Add a data collection rule** on the **Connectors** page of an edge host.)



2. Select **Windows Event Log**.
3. Configure the collection rule.
    1. Enter a name for the collection rule into the **Collection rule name** field.



2. (Optional) Add labels to the collection rule.

   AxoConsole automatically adds the following selectors based on the collector type:

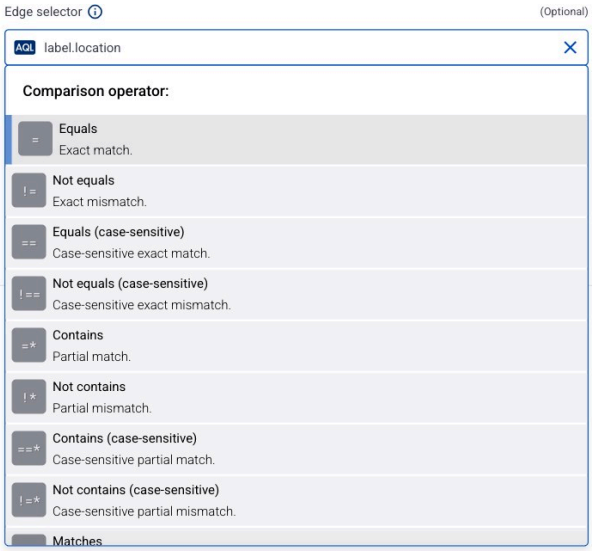
   - **File Collector**: `vendor = opentelemetry` `product = otel-file`
   - **Journald Collector**: `vendor = opentelemetry` `product = otel-journald`
   - **Windows Event Log**: `vendor = microsoft` `product = windows-event-log`
   - **Windows Event Tracing**: `vendor = microsoft` `product = windows-event-tracing`

   You can use these metrics labels as:

   - labels in **Customize Graph** on the Analytics page
   - in the **Filter labels** field during log tapping
   - in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps.

   For edge-related metrics, see the metrics beginning with `edge_connector`

3. Set the **Edge selector** for the collection rule. The selector determines which edge hosts will have a collector based on this collection rule.



   - Only edge hosts will match the rule.
   - If you leave the edge selector empty, the rule matches every edge host.

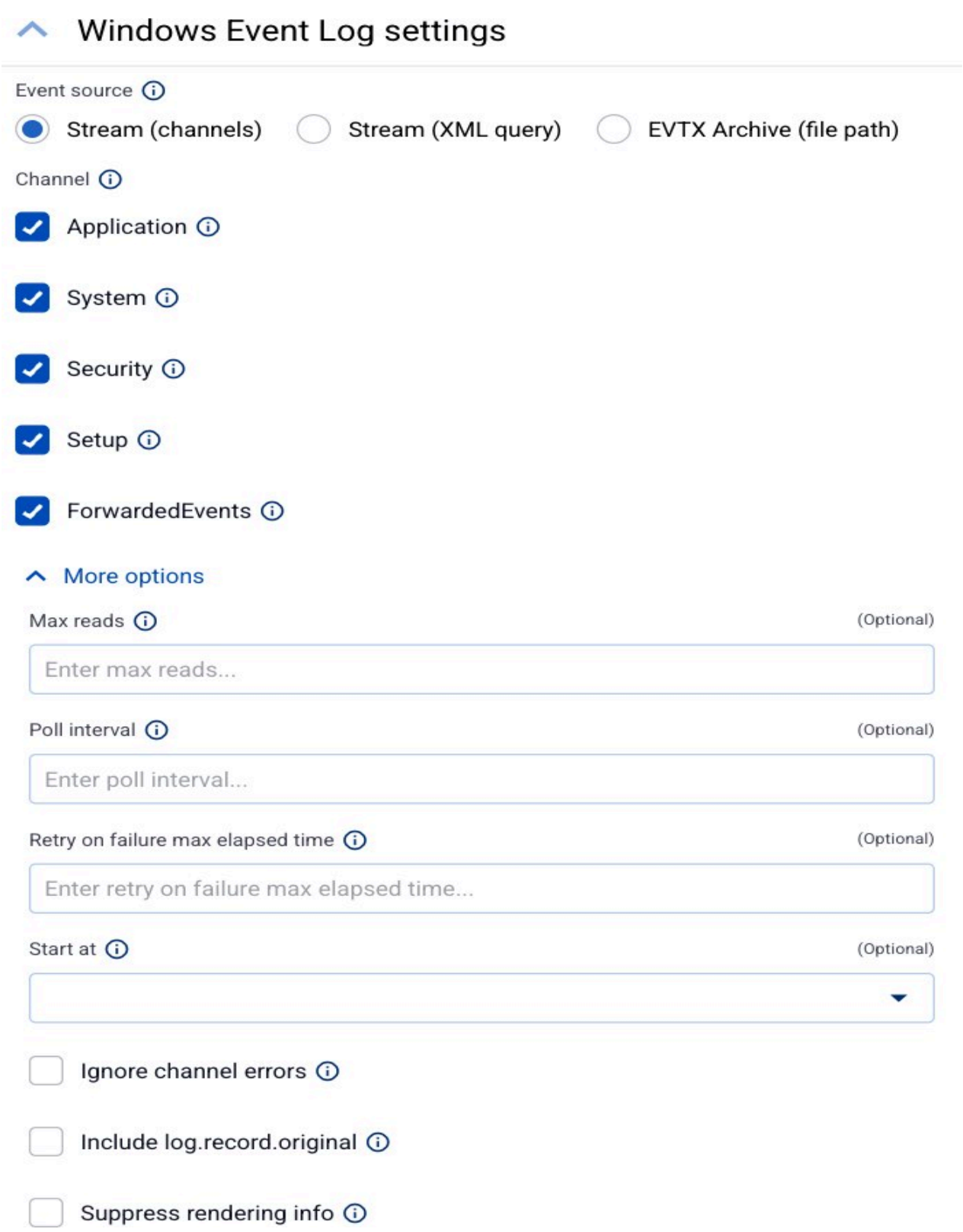
- To select only a specific host, set the `name` field to the name of the host as selector.
- If you set multiple fields in the selector, the collection rule will apply only to edge hosts that match all elements of the selector. (There in an AND relationship between the fields.) For example, `label.location = us-east-1 AND label.product = windows`
- AxoConsole automatically adds the following selectors based on the collector type:
  - **Journald Collector**: `label.platform = linux`
  - **Windows Event Log**: `label.platform = windows`
  - **Windows Event Tracing**: `label.platform = windows`

4. (Optional) Enter a **Suffix** for the collection rule. AxoConsole adds the suffix to the name of the collector instances that it creates on the edge hosts. For example, if the name of a matching edge host is `edge-host-1`, and the suffix of the rule is `otel-file-collector`, the name of the collector on that host is `edge-host-1-otel-file-collector`.

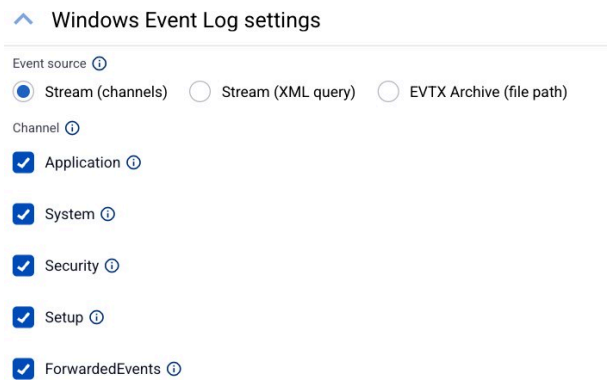
   If the **Suffix** field is empty, AxoConsole uses the name of the collection rule instead.

5. (Optional) Enter a description for the rule.

4. Select which Event Logs to collect.

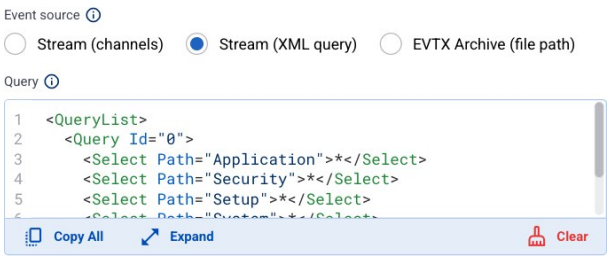


- To collect all Event Logs from the selected built-in channels, select **Stream (channels)**, then the channels to collect data from: **Application**, **System**, **Security**, **Setup**, and **ForwardedEvents**.

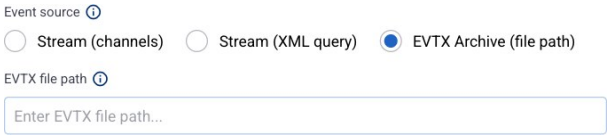


- To use a custom query, select **Stream (XML query)**, then enter the event filter query into the **Query** field. This query specifies which events are collected by the subscription. For details on the query syntax, see the Microsoft documentation. A single query can retrieve events from a maximum of 256 different channels. For example, the following example queries every event from the Application, Security, System, and Setup channels.

```
<QueryList>
  <Query Id="0">
    <Select Path="Application">*</Select>
    <Select Path="Security">*</Select>
    <Select Path="Setup">*</Select>
    <Select Path="System">*</Select>
  </Query>
</QueryList>
```



- To collect data from an EVTX file, select **EVTX Archive (file path)**, then enter the path to the file to read, for example `%SystemRoot%\System32\example.evtx` or `D:\example.evtx`.



5. (Optional) If needed, set advanced options under **More options**.

6. Select **Add**. Based on the collection rule, Axoflow automatically creates collectors on the edge hosts that match the **Edge selector**.

> **CAUTION:**
>
> Make sure to configure data forwarding rules for your edge hosts to transfer the collected data to the OpenTelemetry connector of an AxoRouter.

Related metrics

You can use these metrics labels as:

- labels in **Customize Graph** on the Analytics page
- in the **Filter labels** field during log tapping
- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps.

label	Value
edge_connector_name	The name of the collector that collected the message.
edge_connector_type	`windowsEventLog`
edge_connector_label_	Labels set by the collector. By default: `vendor:microsoft` , `product:windows-event-log`
edge_connector_rule_id	The ID of the Collector Rule resource in Axoflow that created the collector.
edge_flow_name	The name of the data forwarding rule that sent the message.

Advanced options

- **Max reads**: The maximum number of records to read, before beginning a new batch.
- **Poll interval**: The duration between filesystem polls, for example: `1s` , `5m` , `1h` . Default value: `200ms`
- **Retry on failure max elapsed time**: Maximum time (including retries) to send a log batch to a downstream consumer before discarding it, for example: `1s` , `5m` , `1h` . Retrying never stops if set to `0` . Default value `0`
- **Start at**: Specifies where to start reading logs on startup: `beginning` or `end` of the file. Default value: `beginning`

- **Ignore channel errors**: If enabled, the connector keeps working if it cannot open an event log channel.
- **Raw**: If disabled, the body of the emitted log records will contain a structured representation of the event. If enabled, the body will be the original XML string.
- **Include log.record.original**: If enabled, `log.record.original` is added to the attributes of the event. This stores the original XML string as configured in **Suppress rendering info**.
- **Suppress rendering info**: If disabled, additional syscalls may be made to retrieve detailed information about the event. If enabled, some unresolved values may be present in the event.

7.2.4 Windows Event Tracing

Collect Event Tracing for Windows (ETW) events from Windows edge hosts, and look up the labels, metrics, and advanced options.

Collect logs from Event Tracing for Windows (ETW).

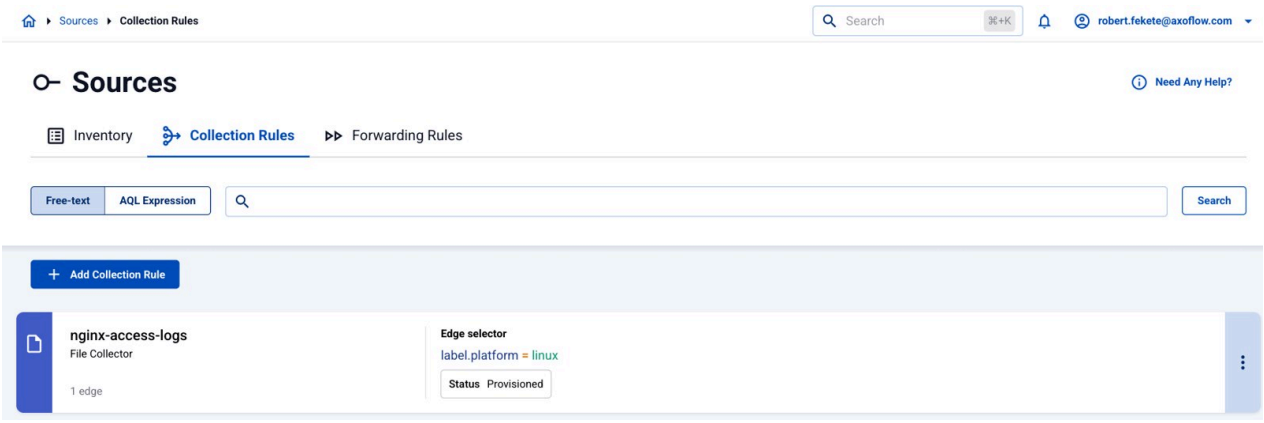
Prerequisites

This collector can be deployed to edge hosts running Axoflow agent for Windows.

Add new ETW Collector

To create a new collection rule that collects Event Tracing data from edge hosts, complete the following steps:

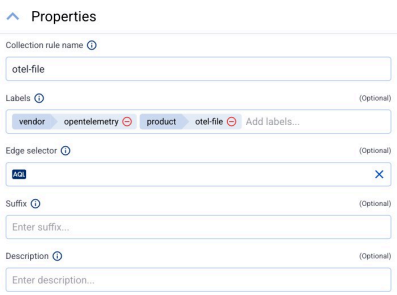
1. Select **Sources > Collection Rules > Add Collection Rule**. (Alternatively, you can select **Add Connector > Add a data collection rule** on the **Connectors** page of an edge host.)



2. Select **Windows Event Tracing**.

3. Configure the collection rule.

    1. Enter a name for the collection rule into the **Collection rule name** field.



2. (Optional) Add labels to the collection rule.

   AxoConsole automatically adds the following selectors based on the collector type:

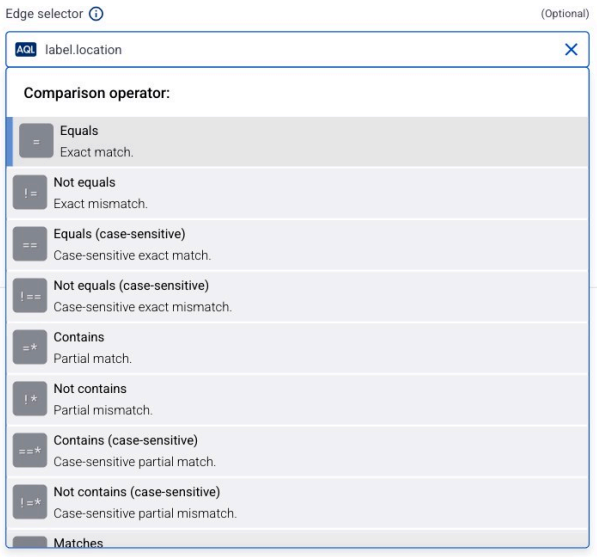
   - **File Collector**: `vendor = opentelemetry` `product = otel-file`
   - **Journald Collector**: `vendor = opentelemetry` `product = otel-journald`
   - **Windows Event Log**: `vendor = microsoft` `product = windows-event-log`
   - **Windows Event Tracing**: `vendor = microsoft` `product = windows-event-tracing`

   You can use these metrics labels as:

   - labels in **Customize Graph** on the Analytics page
   - in the **Filter labels** field during log tapping
   - in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps.

   For edge-related metrics, see the metrics beginning with `edge_connector`

3. Set the **Edge selector** for the collection rule. The selector determines which edge hosts will have a collector based on this collection rule.



   - Only edge hosts will match the rule.
   - If you leave the edge selector empty, the rule matches every edge host.
   - To select only a specific host, set the `name` field to the name of the host as selector.
   - If you set multiple fields in the selector, the collection rule will apply only to edge hosts that match all elements of the selector. (There in an AND relationship between the fields.) For example, `label.location = us-east-1 AND label.product = windows`
   - AxoConsole automatically adds the following selectors based on the collector type:

- **Journald Collector**: `label.platform = linux`
- **Windows Event Log**: `label.platform = windows`
- **Windows Event Tracing**: `label.platform = windows`

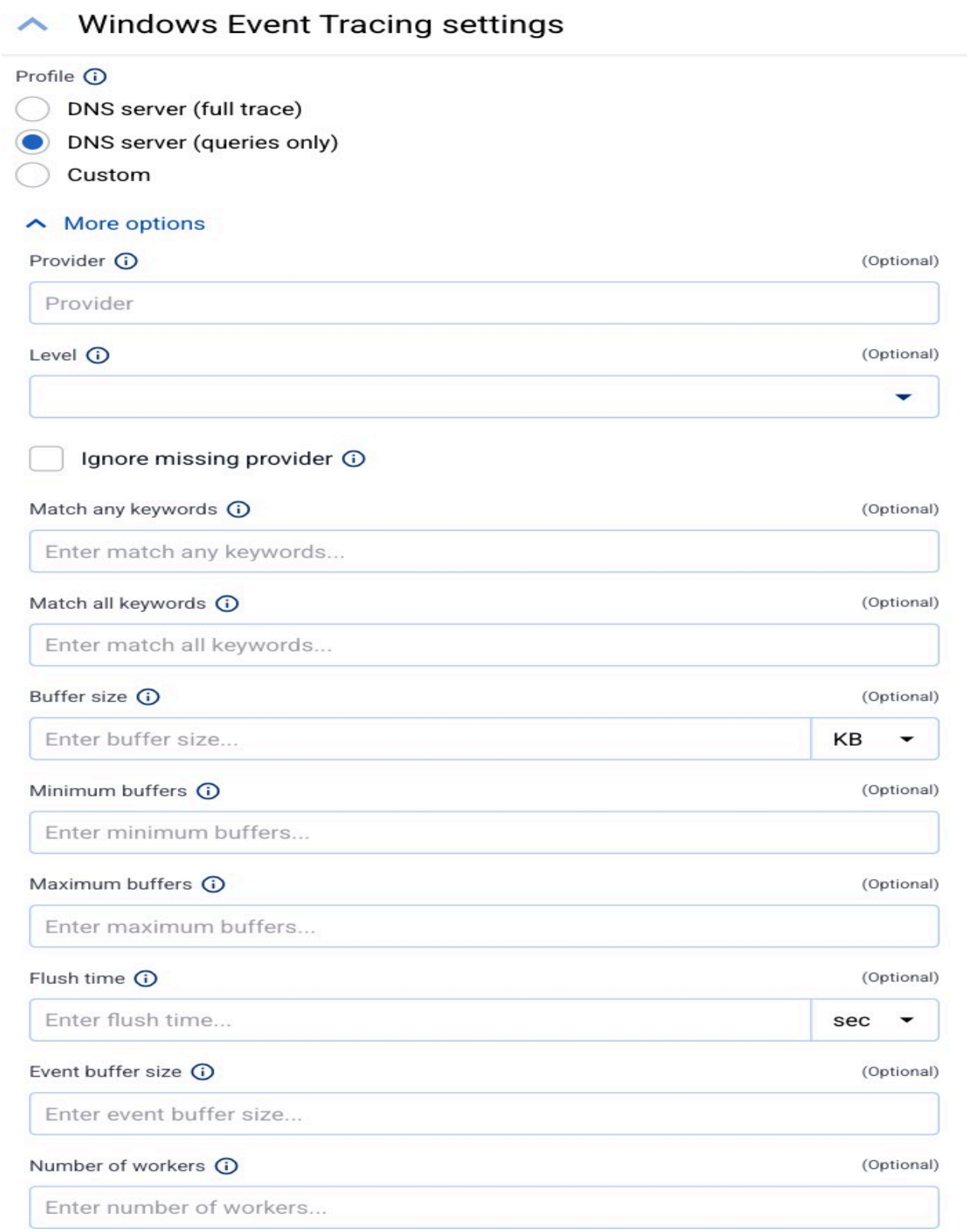
4. (Optional) Enter a **Suffix** for the collection rule. AxoConsole adds the suffix to the name of the collector instances that it creates on the edge hosts. For example, if the name of a matching edge host is `edge-host-1`, and the suffix of the rule is `otel-file-collector`, the name of the collector on that host is `edge-host-1-otel-file-collector`.

   If the **Suffix** field is empty, AxoConsole uses the name of the collection rule instead.

5. (Optional) Enter a description for the rule.

4. Select the configuration profile to use. The following profiles are available:

   - **DNS server (full trace)**: A pre-defined profile for collecting every available DNS server traces.
   - **DNS server (queries only)**: A pre-defined profile for collecting only DNS queries.
   - **Custom**: Use a fully-customized configuration.



5. (Optional) If needed, set advanced options under **More options**.

   Note that if you set an advanced option when using a pre-defined profile, your changes override the related default setting of the pre-defined profile.

6. Select **Add**. Based on the collection rule, Axoflow automatically creates collectors on the edge hosts that match the **Edge selector**.

CAUTION:

Make sure to configure data forwarding rules for your edge hosts to transfer the collected data to the OpenTelemetry connector of an AxoRouter.

Note

When using Log tapping, ETW events look a bit weird: the body of these events is empty. That's normal, the reason for that is that everything is sent as metadata.

Related metrics

You can use these metrics labels as:

- labels in **Customize Graph** on the Analytics page
- in the **Filter labels** field during log tapping
- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps.

label	Value
edge_connector_name	The name of the collector that collected the message.
edge_connector_type	`windowsEventTracing`
edge_connector_label_	Labels set by the collector. By default: `vendor:microsoft` , `product:windows-event-tracing`
edge_connector_rule_id	The ID of the Collector Rule resource in Axoflow that created the collector.
edge_flow_name	The name of the data forwarding rule that sent the message.

Advanced options

Note that if you set an advanced option when using a pre-defined profile, your changes override the related default setting of the pre-defined profile.

- **Provider**: The provider to subscribe to, for example, `Microsoft-Windows-DNSServer` for DNS logs. For a complete list, open a command prompt on the edge host and run `logman query providers` .

Note that provider name is case sensitive. Alternatively, you can use the GUID of the provider as well, in the following format: `{9e814aad-3204-11d2-9a82-006008a86939}`. If you're manually setting the provider, consider enabling the **Ignore missing provider** option as well.

- **Level**: Log level of trace events to be included. Specifying a log level means that all higher priority log levels will be collected as well. Possible values are (starting with the highest priority): `critical`, `error`, `warning`, `information`, `verbose`.
- **Ignore missing provider**: Continue working if the specified provider is missing from the host.

  > **CAUTION:**
  >
  > If this option is disabled and the provider is missing, the all other connectors of the agent can stop, not just the ETW connectors.

- **Match any keywords**: Collect only traces that match at least one of the specified keywords of the provider. Note that these keywords are not literal strings, but bitmasks that correspond to the specific provider. To match on multiple keywords, you have to add the bitmasks of the corresponding keywords.
- **Match all keywords**: Collect only traces that match all of the specified keywords of the provider. Note that these keywords are not literal strings, but bitmasks that correspond to the specific provider. To match on multiple keywords, you have to add the bitmasks of the corresponding keywords.
- **Buffer size**: Buffer size allocated for each ETW trace session, in kilobytes. Minimum is 4, maximum is 16384.
- **Minimum buffers**: Minimum number of buffers allocated for each ETW trace session. Note that the minimum and maximum buffers behave like hints to the ETW subsystem and aren't guaranteed to be allocated as specified.
- **Maximum buffers**: Maximum number of buffers allocated for each ETW trace session. Note that the minimum and maximum buffers behave like hints to the ETW subsystem and aren't guaranteed to be allocated as specified.
- **Flush time**: How often, in seconds, any non-empty trace buffers are flushed. `0` will enable a default timeout of `1` second.
- **Event buffer size**: Number of ETW events the ETW receiver stores in memory for processing.
- **Number of workers**: Number of workers that process ETW events.

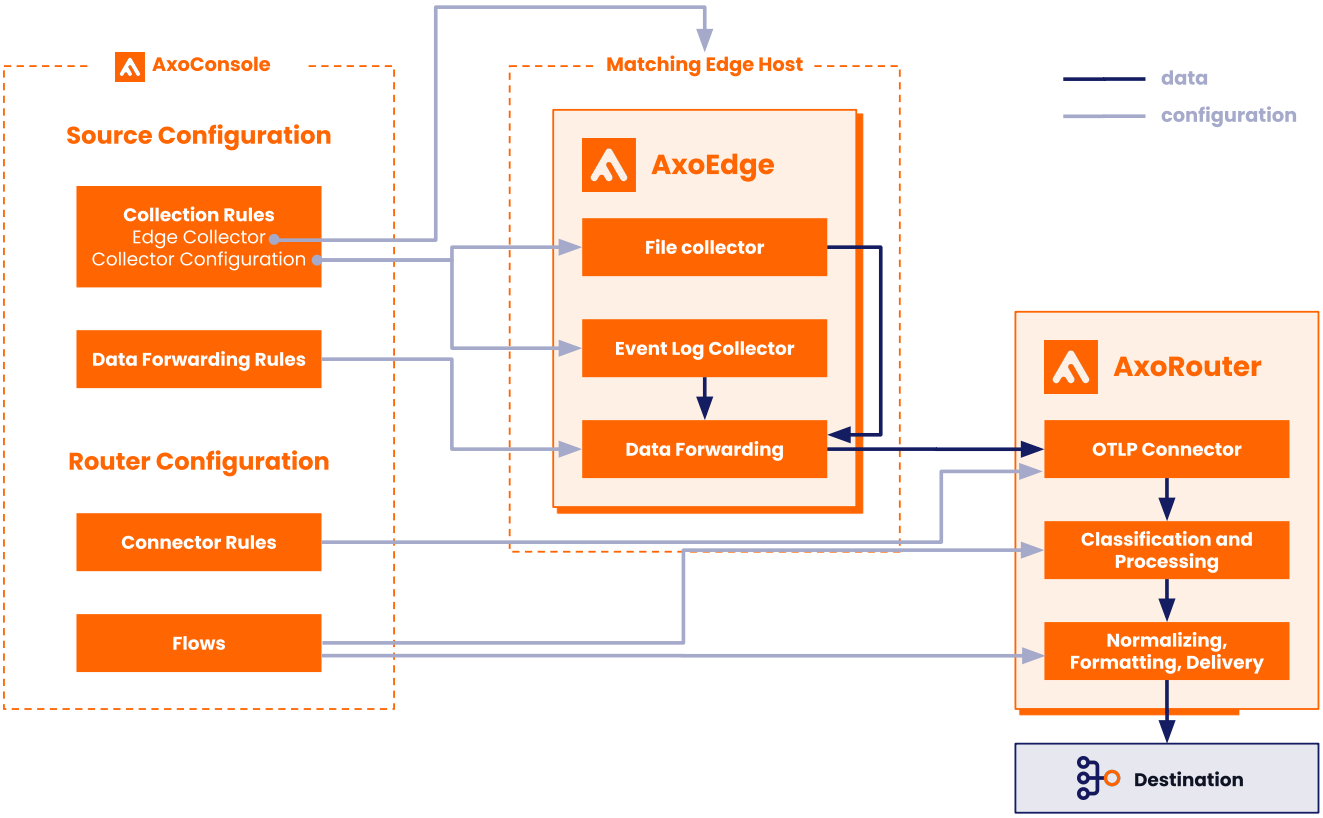
7.3 Data forwarding from edge hosts

Create and modify the forwarding rules that decide which AxoRouter receives the data of your edge hosts.

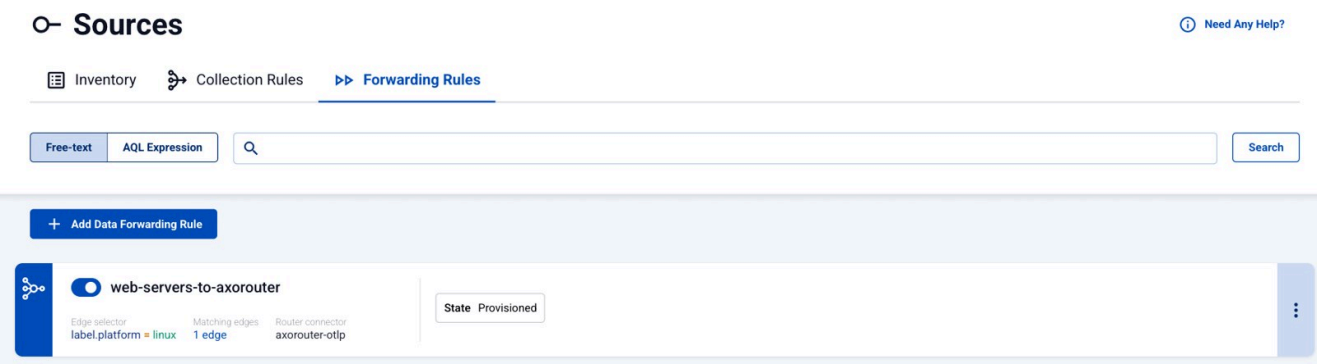
To send data from an edge host to an AxoRouter, create a data forwarding rule.

- Sources are hosts that are sending data to a data aggregator, like AxoRouter.
- Edges are source hosts that are running a collector agent managed by AxoConsole, or have an Axolet agent reporting metrics from the host.

Data forwarding rules are high-level policies that determine how data should be forwarded from a set of edge hosts to an AxoRouter's OpenTelemetry connector. You can use dynamic host labels to specify the edge hosts that the forwarding rule applies to.



To see every data forwarding rule configured in AxoConsole, select **Sources > Forwarding Rules** from the main menu.



Data forwarding rules have the following main elements:

- the edge selector, which determines the list of edge hosts that will create a data forwarding rule based on that rule.

  You can use any labels and metadata of the edge hosts in the edge selectors, for example, the hostname, or any custom labels. For example, using the `label.product = windows` selector will create a data forwarding rule only on Windows hosts.

- the Router connector to send the data to. This must be an OpenTelemetry connector of an AxoRouter.

Selecting the name of a data forwarding rule opens its **Overview** page, which shows the details of the rule, including the list of **Matching Edges**. To turn a rule off or on, use the toggle next to its name.

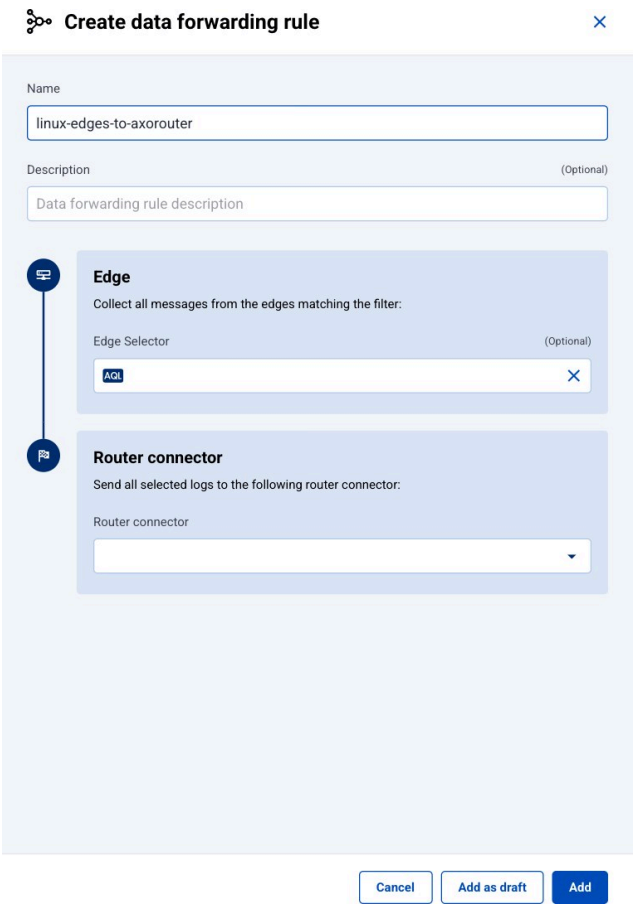
Create data forwarding rule

To create a new data forwarding rule, complete the following steps.

1. If the primary IP address (the first IP address shown in the **Network addresses** section on the **Routers** page for each AxoRouter) is not accessible from your edge hosts, set a **Network address override** (IP address or an FQDN) that's accessible. Otherwise, data forwarding from edge hosts will fail.

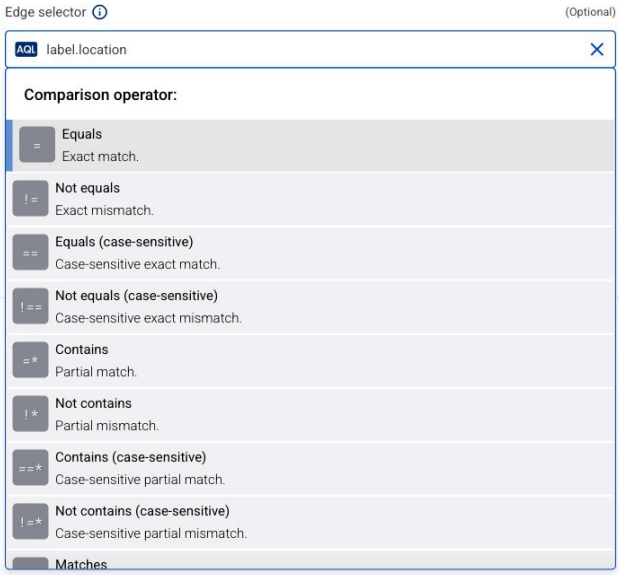
2. Select **Sources > Forwarding Rules > Add Data Forwarding Rule**.

3. Enter a name for the data forwarding rule into the **Name** field.



4. (Optional) Enter a description for the rule.

5. Set the **Edge Selector** for the rule. The selector determines which edge hosts will have data forwarding configured based on this rule.



   - Only edge hosts will match the rule.
   - If you leave the edge selector empty, the rule matches every edge host.
   - To select only a specific host, set the `name` field to the name of the host as selector.

- If you set multiple fields in the selector, the collection rule will apply only to edge hosts that match all elements of the selector. (There in an AND relationship between the fields.) For example, `label.location = us-east-1 AND label.product = windows`

6. In the **Router connector** field, select the OpenTelemetry connector of an AxoRouter where you want to send the data.

7. Select **Add** to create and immediately apply the data forwarding rule to the matching edge hosts. Alternatively, select **Add as draft** to create the rule in a disabled state. You can enable the rule later on the **Sources > Forwarding Rules** page, with the toggle next to the name of the rule.

Modify forwarding rule

To modify an existing data forwarding rule, complete the following steps.

> **Note**
>
> - You cannot directly modify the data forwarding settings of an edge host, only via modifying a data forwarding rule.
> - Modifying a rule affects every edge host that matches the **Edge Selector** of the rule.
> - To apply an existing rule to a new edge host, you can:
>   - Add a label to the edge host so the **Edge Selector** of the rule matches the new host, and/or
>   - Change the **Edge Selector** of the rule to match the new host.

1. Select **Sources > Forwarding Rules** from the main menu, then select the name of the rule you want to modify.

2. Select **Process**, then modify the **Edge Selector** or the **Router connector** of the rule as needed. (To change the description of the rule, select **Edit** instead.)

   > **CAUTION:**
   >
   > The changes are applied immediately after you select **Save**. Double-check your changes to avoid losing data.

3. Select **Save**.

Add edge host to existing forwarding rule

To add an edge host to an existing data forwarding rule, you have two options, depending on the **Edge Selector** of the data forwarding rule:

- Modify the **Edge Selector** field of the rule to include the new edge host.
- Modify the labels of the edge host so it matches the edge selector of the rule.

Troubleshooting

If the data doesn't arrive at the destination, see An edge host doesn't send data. For other problems, see Troubleshooting Axoflow.

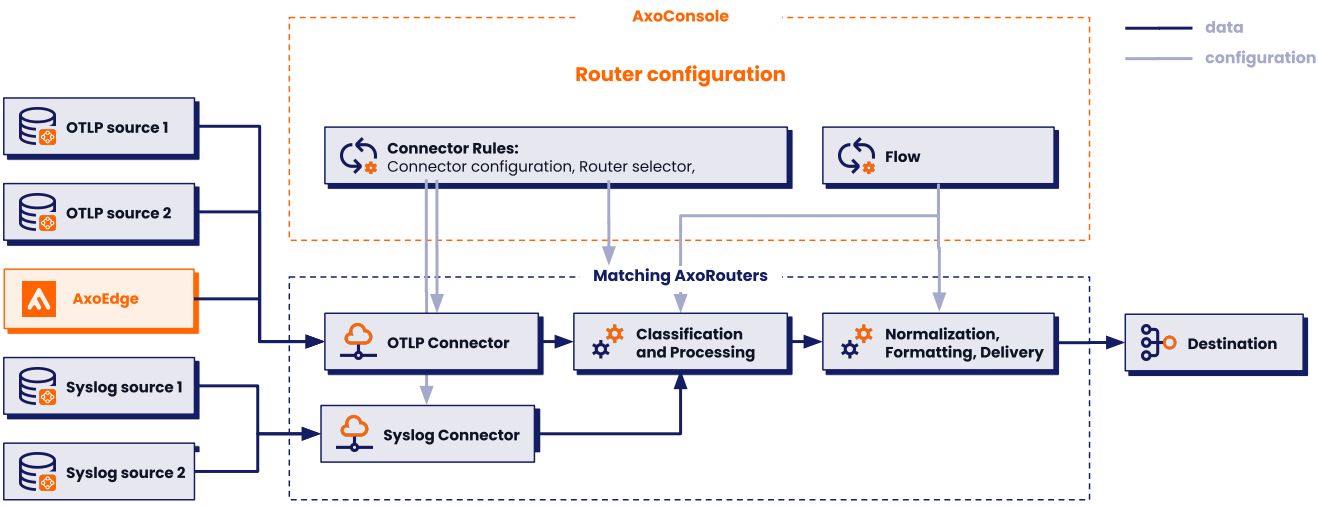
7.4 AxoRouter connector rules

Create and modify connector rules that set which listeners your AxoRouter instances open, and how they classify and parse the received data.

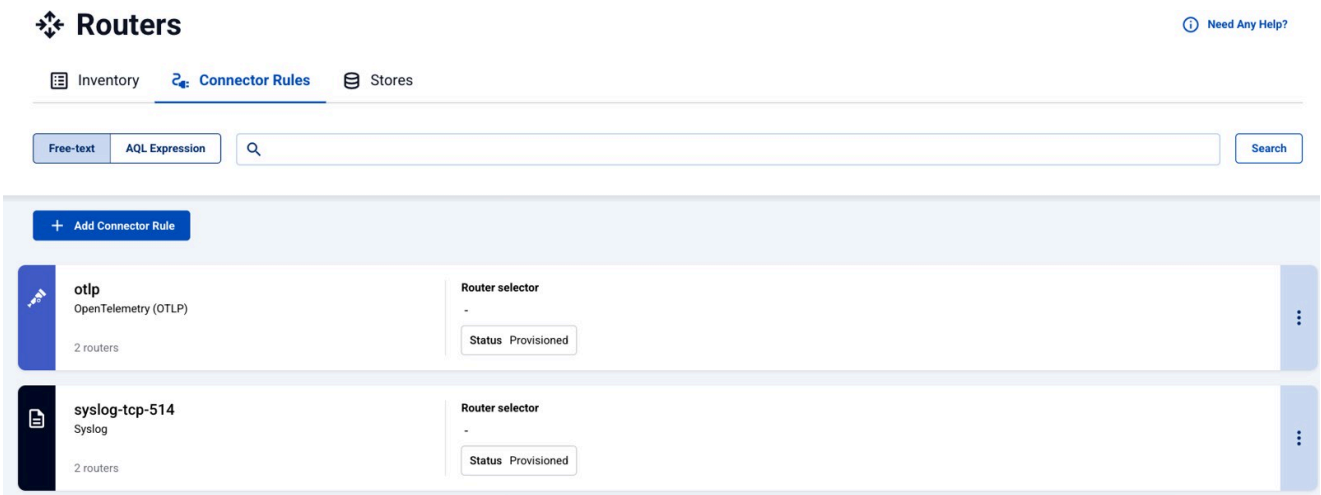
Connectors define the methods and entry points where AxoRouter receives data from the sources. To collect data from edge hosts, see Edge collection rules.

In addition to providing a simple protocol specific listener, Connectors implement critical processing steps - for example classification and parsing - right at the ingestion point to enrich data before it gets processed by Flows.

On top of this, Connector rules are high-level policies that determine what kind of Connectors should be available on a set of AxoRouter instances based on dynamic host labels.



To see every connector rule configured in AxoConsole, select **Routers > Connector Rules** from the main menu.



Connector rules have the following main elements:

- the way they accept data (for example, via syslog, using the OpenTelemetry protocol), and other protocol-specific parameters (for example, port number)
- the router selector, which determines the list of AxoRouter instances that will create a connector based on that rule.

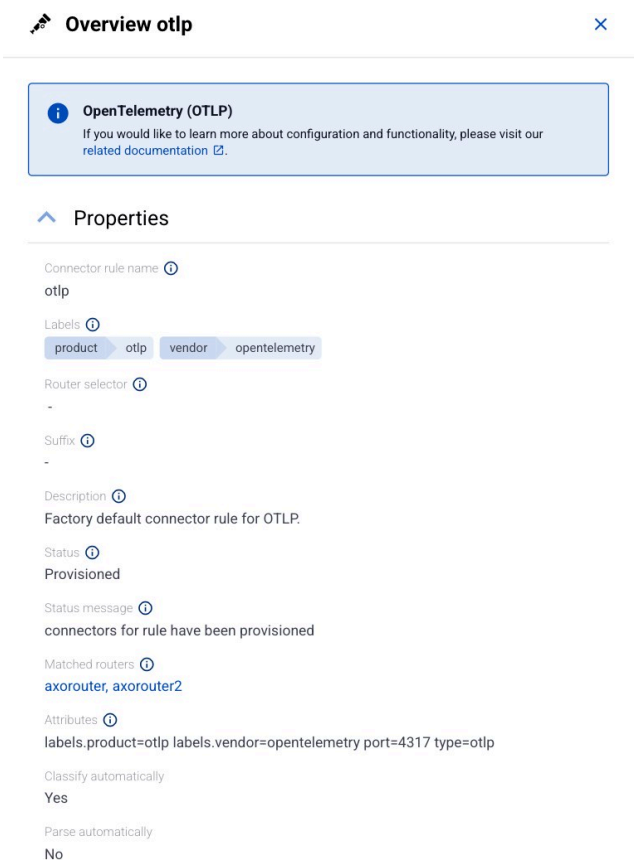
  You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

  - If you leave the router selector empty, it matches every AxoRouter instance.
  - To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.
  - If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

  For example, using the `label.wec = enabled` selector on a Windows Event Collector (WEC) rule will create a WEC connector on AxoRouter hosts that have the `wec` label set to `enabled`.

To show the details of a connector rule, select more_vert > **Details** on its card. The details include:

- The list of **Matched routers**: the AxoRouter deployments that will have a connector based on that connector rule. If you select the name of a matched router, the **Connectors** page of the AxoRouter host opens, showing you the connectors configured for that host.
- **Classify automatically** and **Parse automatically**: whether the **Classify** and **Parse** preprocessing steps of the rule are turned on.

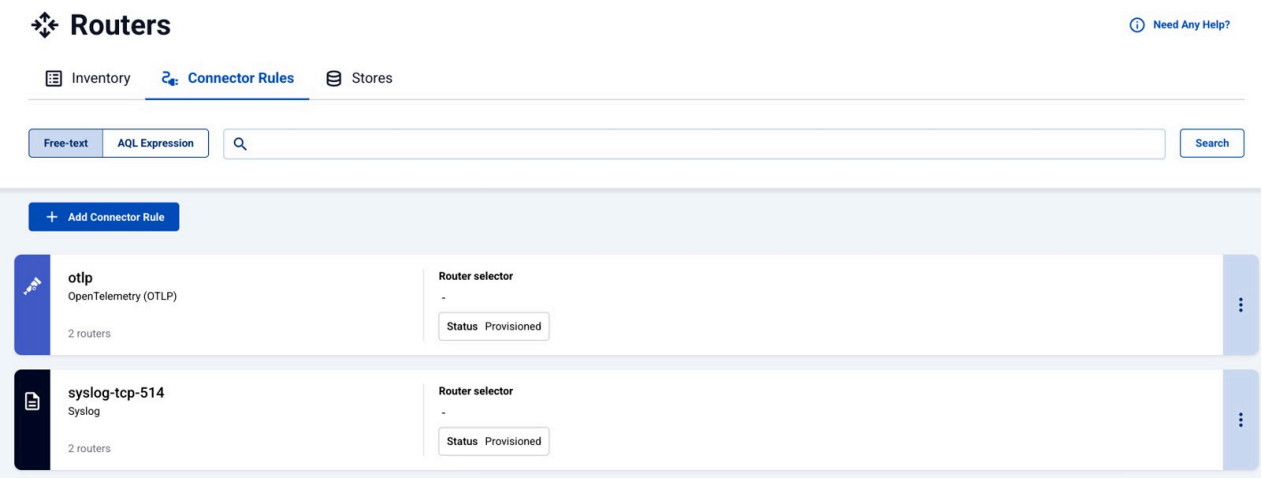


- The **Status** of the rule:
  - **Idle**: The rule doesn't match any hosts currently.
  - **Provisioned**: Connectors based on this rule were successfully provisioned for every matching host.
  - **Error**: Some error(s) occurred while provisioning connectors based on this rule. See the **Status message** field for details.
  - **Unknown**: The rule is in an unknown state.
- **Attributes**: Various significant attributes of connectors provisioned based on this rule.

Create connector rule

To create a new connector rule, complete the following steps.

1. Select **Routers > Connector Rules > Add Connector Rule**. (Alternatively, you can select **Add Connector > Add a connector rule** on the **Connectors** page of an AxoRouter host.)

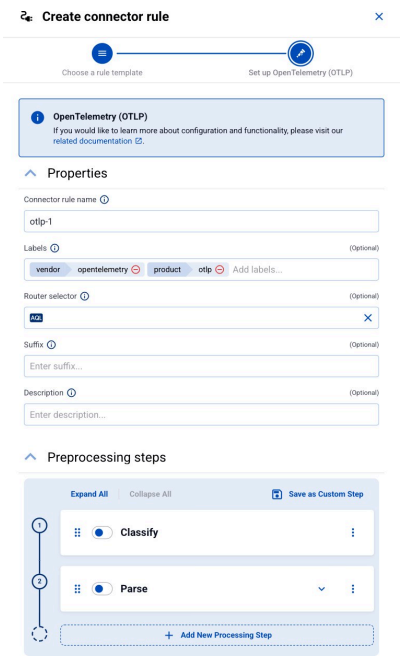


2. Select the type of connector you want to create. For example, **OpenTelemetry**. The following connector types are available:

   - AWS CloudTrail
   - AWS VPC Flow Logs
   - Azure Event Hubs
   - Elasticsearch Bulk API
   - Falcon Platform (CrowdStrike)
   - Generic HTTP
   - Idira (Palo Alto Networks)
   - OpenTelemetry (OTLP)
   - Splunk HEC Server
   - Syslog
   - Vulnerability Management (Tenable)
   - Webhook
   - Windows Events (Windows Event Collector)

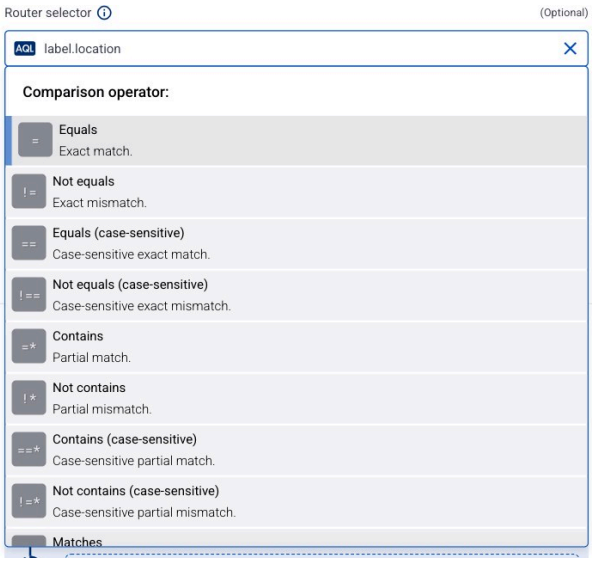
3. Configure the connector rule.

   1. Enter a name for the connector rule into the **Connector rule name** field.



2. (Optional) Add labels to the connector rule. You will be able to use these labels in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps. For details about the message schema and the related connector-specific fields, see the `meta.connector` object.

3. Set the **Router selector** for the connector rule. The selector determines which AxoRouter instances will have a connector based on this connector rule.



You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

- If you leave the router selector empty, it matches every AxoRouter instance.
- To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.

- If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

4. (Optional) Enter a **Suffix** for the connector rule. AxoConsole uses the suffix in the name of the connector instances that it creates on the AxoRouter hosts. The name of the connector instance follows the `<name-of-AxoRouter>-<suffix>` schema. For example, if the name of a matching AxoRouter instance is `axorouter-1`, and the suffix of the rule is `custom`, the name of the connector on that AxoRouter is `axorouter-1-custom`.

   If the **Suffix** field is empty, AxoConsole uses the name of the connector rule instead. For example, if the name of the connector rule is `otlp-1`, the name of the connector is `axorouter-1-otlp-1`.

5. (Optional) Enter a description for the rule.

6. If needed, turn on the **Classify** and **Parse** steps in the **Preprocessing steps** section (they're off by default), so AxoRouter automatically identifies and parses messages sent by supported data sources. If your source is not listed, contact us.

   Enabling these options processes all data received by the connectors created based on this connector rule. If you want to apply classification and parsing more selectively, you can use the **Classify** and **Parse** processing steps in your **Flows**.

   Note that the **Parse** processing step requires **Classify** to be enabled. Parsing automatically parses the data from the content of the message, and replaces the message content (the `log.body` field in the internal message schema) with the structured information.

7. (Optional) If needed, you can add other preprocessing steps to the connector rule. You can use the same processing steps as in flows. These can be useful if you use a dedicated connector rule to fix or annotate data coming from some specific sources. To reuse the preprocessing steps in other connector rules or flows, select **Save as Custom Step**.

4. Configure the protocol-specific settings. For details, see the specific pages:

   - AWS CloudTrail
   - AWS VPC Flow Logs
   - Azure Event Hubs
   - Elasticsearch Bulk API
   - Falcon Platform (CrowdStrike)
   - Generic HTTP

- Idira (Palo Alto Networks)
- OpenTelemetry (OTLP)
- Splunk HEC Server
- Syslog
- Vulnerability Management (Tenable)
- Webhook
- Windows Events (Windows Event Collector)

> **CAUTION:**
>
> TCP and UDP port range 29500–29519 is reserved for AxoRouter's internal listeners and cannot be used in connector rules.

5. Select **Add**.

   Axoflow automatically creates connectors on the AxoRouter hosts that match the **Router selector**.

   Make sure to enable the ports you've configured in the connector on the firewall of the AxoRouter host, and on other firewalls between the AxoRouter host and your data sources.

Modify connector rule

To modify an existing connector rule, complete the following steps.

> **Note**
>
> - You cannot directly modify the connector of an AxoRouter host, only via modifying a connector rule.
> - Modifying a connector rule affects every AxoRouter that matches the **Router selector** of the rule.
> - To apply an existing connector rule to a new AxoRouter instance, you can:
>   - Add a label to the AxoRouter host so the **Router selector** of the connector rule matches the new host, and/or
>   - Change the **Router selector** of the connector rule to match the new host.

1. Find the connector rule you want to modify:

   - Select **Routers > Connector Rules** from the main menu, then select more_vert > **Edit** on the card of the connector rule you want to modify.

- Alternatively, find the AxoRouter instance whose connector you want to modify on the **Topology** page, then select **Connectors**. Find the connector you want to modify, then select more_vert > **Edit connector rule**.

2. Modify the configuration of the connector rule as needed.

   > **CAUTION:**
   >
   > The changes are applied immediately after you click **Update**. Double-check your changes to avoid losing data.

3. Select **Update**.

Add AxoRouter to existing connector rule

To add an AxoRouter to an existing connector rule, you have two options, depending on the **Router selector** of the connector rule:

- Modify the **Router selector** field of the connector rule to include the new AxoRouter host.
- Modify the labels of the AxoRouter host so it matches the router selector of the connector rule.

7.5 OpenTelemetry

Add an OpenTelemetry connector to AxoRouter to receive OTLP data from OpenTelemetry clients and Axoflow components.

Receive logs, metrics, and traces from OpenTelemetry clients over the OpenTelemetry Protocol (OTLP/gRPC).

Prerequisites

If you want to enable TLS encryption for this connector to encrypt the communication with the sources, you'll need to set appropriate keys and certificates.

> **CAUTION:**
>
> Copy the keys and certificates to AxoRouter before starting to configure the connector. Otherwise, you won't be able to make configuration changes that require reloading the AxoRouter service, including starting log tapping or flow tapping.

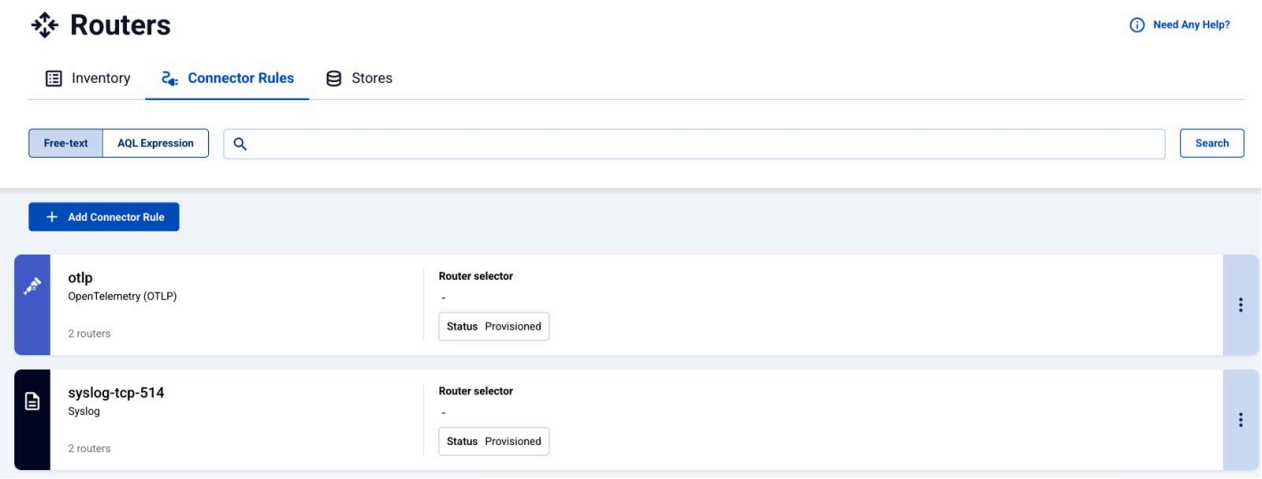
Note the following points:

- Keys and certificates must be in PEM format.
- If the file contains a certificate chain, the file must begin with the certificate of the host, followed by the CA certificate that signed the certificate of the host, and any other signing CAs in order.
- If you manually copy these files to their place on the AxoRouter host, make the files readable by the `axorouter` service.
  - The recommended path for certificates is under `/etc/axorouter/user-config/` (for example, `/etc/axorouter/user-config/tls-key.pem`). (If you need to use a different path, you have to append an option like `-v /your/path:/your/path` to the `AXOROUTER_PODMAN_ARGS` variable of `/etc/axorouter/container.env`.)
  - When referring to the key or certificate when configuring the connector, use absolute paths (for example, `/etc/axorouter/user-config/tls-key.pem`).

Add new OpenTelemetry connector

To add a new connector to an AxoRouter host, complete the following steps:

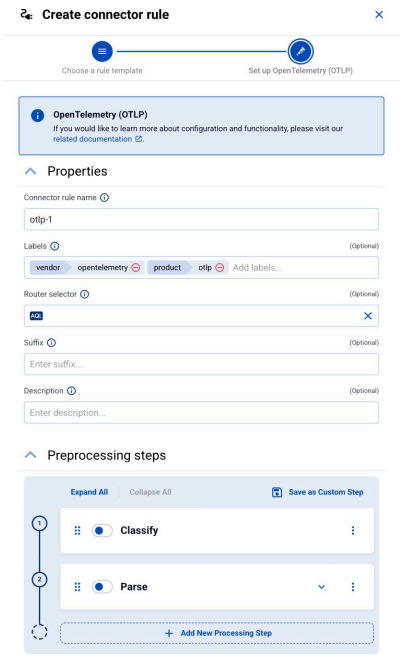
1. Select **Routers > Connector Rules > Add Connector Rule**. (Alternatively, you can select **Add Connector > Add a connector rule** on the **Connectors** page of an AxoRouter host.)



2. Select **OpenTelemetry (OTLP)**.

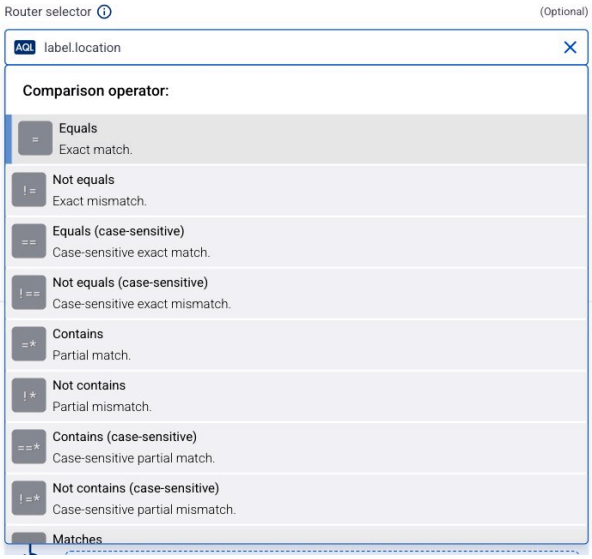
3. Configure the connector rule.

    1. Enter a name for the connector rule into the **Connector rule name** field.



    2. (Optional) Add labels to the connector rule. You will be able to use these labels in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps. For details about the message schema and the related connector-specific fields, see the `meta.connector` object.

    3. Set the **Router selector** for the connector rule. The selector determines which AxoRouter instances will have a connector based on this connector rule.



You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

- If you leave the router selector empty, it matches every AxoRouter instance.
- To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.
- If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

4. (Optional) Enter a **Suffix** for the connector rule. AxoConsole uses the suffix in the name of the connector instances that it creates on the AxoRouter hosts. The name of the connector instance follows the `<name-of-AxoRouter>-<suffix>` schema. For example, if the name of a matching AxoRouter instance is `axorouter-1`, and the suffix of the rule is `custom`, the name of the connector on that AxoRouter is `axorouter-1-custom`.

   If the **Suffix** field is empty, AxoConsole uses the name of the connector rule instead. For example, if the name of the connector rule is `otlp-1`, the name of the connector is `axorouter-1-otlp-1`.

5. (Optional) Enter a description for the rule.

6. If needed, turn on the **Classify** and **Parse** steps in the **Preprocessing steps** section (they're off by default), so AxoRouter automatically identifies and parses messages sent by supported data sources. If your source is not listed, contact us.

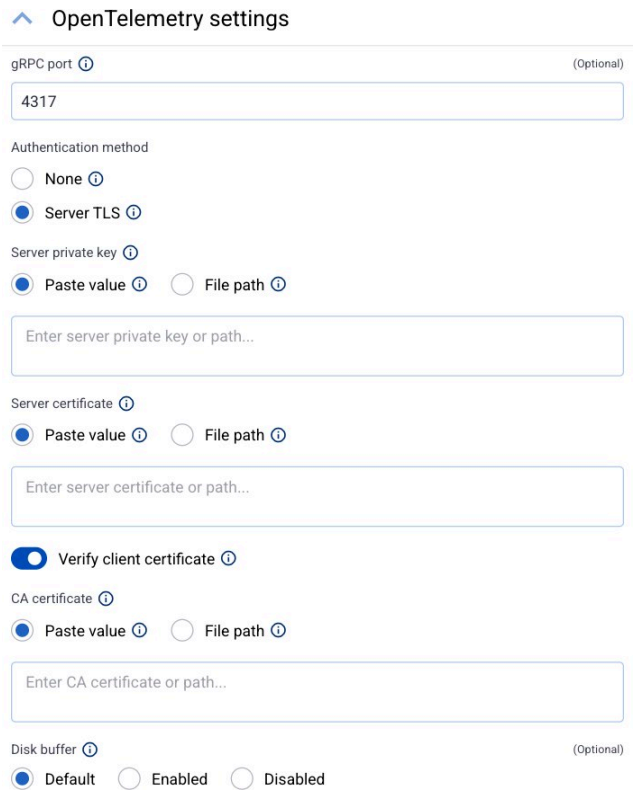
   Enabling these options processes all data received by the connectors created based on this connector rule. If you want to apply classification and parsing more selectively, you can use the **Classify** and **Parse** processing steps in your **Flows**.

Note that the **Parse** processing step requires **Classify** to be enabled. Parsing automatically parses the data from the content of the message, and replaces the message content (the `log.body` field in the internal message schema) with the structured information.

7. (Optional) If needed, you can add other preprocessing steps to the connector rule. You can use the same processing steps as in flows. These can be useful if you use a dedicated connector rule to fix or annotate data coming from some specific sources. To reuse the preprocessing steps in other connector rules or flows, select **Save as Custom Step**.

4. If needed, configure the port number where you want to receive data.

> **CAUTION:**
>
> TCP and UDP port range 29500–29519 is reserved for AxoRouter's internal listeners and cannot be used in connector rules.



5. Configure TLS settings for the connector.

   1. Select **Server TLS**.

   2. Set the path to the key and certificates.

      When using TLS, set the paths for the certificates and keys used for the TLS-encrypted communication with the clients. For details, see Prerequisites.

> **Note**
>
> **These settings apply to every AxoRouter that matches the Router selector of the connector rule. If you paste values for the server certificate and server private key, every matching AxoRouter uses the same shared certificate, which is not recommended. Instead, use a separate certificate for each AxoRouter by setting the file path option, or create a separate connector rule for each AxoRouter.**

- **Server private key**: The private key of the server certificate.
  - To provide the server private key from the AxoConsole, select **Paste value** and paste the server private key. AxoConsole will automatically deploy the server private key to the related AxoRouters.
  - To provide the server private key from a file, select **File path** and enter the path where the private key file is located (for example, `/etc/axorouter/user-config/server-private-key.pem`). The path must be mounted into the container, `/etc/axorouter` is mounted by default. You must manually copy that file to its place on every affected AxoRouter, or you can use the configuration management system of your choice.
- **Server certificate**: The certificate that AxoRouter shows to the clients.
  - To provide the server certificate from the AxoConsole, select **Paste value** and paste the server certificate. AxoConsole will automatically deploy the server certificate to the related AxoRouters.
  - To provide the server certificate from a file, select **File path** and enter the path where the certificate file is located (for example, `/etc/axorouter/user-config/server-certificate.pem`). The path must be mounted into the container, `/etc/axorouter` is mounted by default. You must manually copy that file to its place on every affected AxoRouter, or you can use the configuration management system of your choice.
- **CA certificate**: The CA certificate that AxoRouter uses to verify the certificate of the sender if **Verify client certificate** is enabled.
  - To provide the CA certificate from the AxoConsole, select **Paste value** and paste the CA certificate. AxoConsole will automatically deploy the CA certificate to the related AxoRouters.
  - To provide the CA certificate from a file, select **File path** and enter the path where the certificate file is located (for example, `/etc/axorouter/user-config/ca-certificate.pem`). The path must be mounted into the container, `/etc/axorouter` is mounted by default. You must manually copy that file to its place on every affected AxoRouter, or you can use the configuration management system of your choice.

3. To require the certificates of the clients to be valid, select **Verify client certificate**. When selected, the certificate of the client cannot be self-signed, and its common name must match the hostname or IP address of the client.

6. Select **Add**.

   Axoflow automatically creates connectors on the AxoRouter hosts that match the **Router selector**.

   Make sure to enable the ports you've configured in the connector on the firewall of the AxoRouter host, and on other firewalls between the AxoRouter host and your data sources.

Labels

Analytics label	Message field	Value
`connector.type`	`meta.connector.type`	`otlp`
`connector.name`	`meta.connector.name`	The **Name** of the connector
`connector.port`	`meta.connector.port`	The port number where the connector receives data

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

7.6 Syslog

Add syslog connectors to AxoRouter on the ports and protocols your sources use, and look up the protocol-specific options and labels.

The **Syslog** connector can receive all kinds of syslog messages. You can configure it to receive data on specific ports, and apply parsing, classification, and enrichment to the messages, in addition to standard syslog parsing.

Prerequisites

If you want to enable TLS encryption for this connector to encrypt the communication with the sources, you'll need to set appropriate keys and certificates.

> **CAUTION:**
>
> Copy the keys and certificates to AxoRouter before starting to configure the connector. Otherwise, you won't be able to make configuration changes that require reloading the AxoRouter service, including starting log tapping or flow tapping.

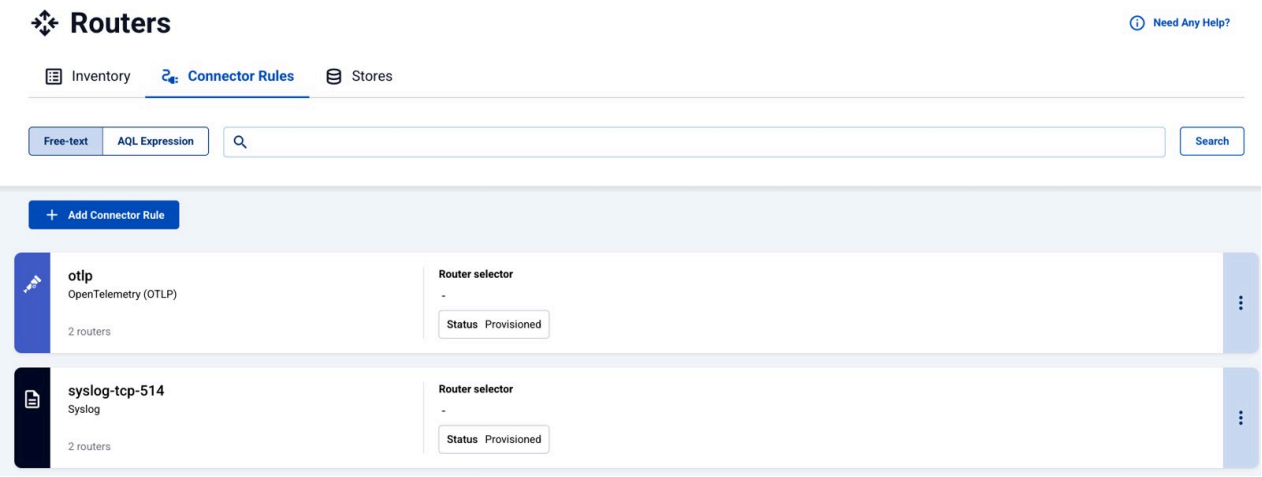
Note the following points:

- Keys and certificates must be in PEM format.
- If the file contains a certificate chain, the file must begin with the certificate of the host, followed by the CA certificate that signed the certificate of the host, and any other signing CAs in order.
- If you manually copy these files to their place on the AxoRouter host, make the files readable by the `axorouter` service.
  - The recommended path for certificates is under `/etc/axorouter/user-config/` (for example, `/etc/axorouter/user-config/tls-key.pem`). (If you need to use a different path, you have to append an option like `-v /your/path:/your/path` to the `AXOROUTER_PODMAN_ARGS` variable of `/etc/axorouter/container.env`.)
  - When referring to the key or certificate when configuring the connector, use absolute paths (for example, `/etc/axorouter/user-config/tls-key.pem`).

Add new syslog connector

To create a new syslog connector, complete the following steps:

1. Select **Routers > Connector Rules > Add Connector Rule**. (Alternatively, you can select **Add Connector > Add a connector rule** on the **Connectors** page of an AxoRouter host.)

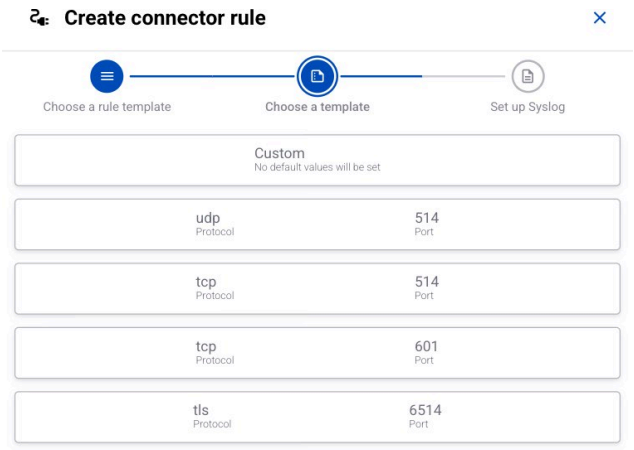


2. Select **Syslog**.

3. Select the template to use one of the standard syslog ports and networking protocols: UDP 514 (for example, for the RFC3164 syslog protocol), TCP 514, TCP 601, or TLS 6514.

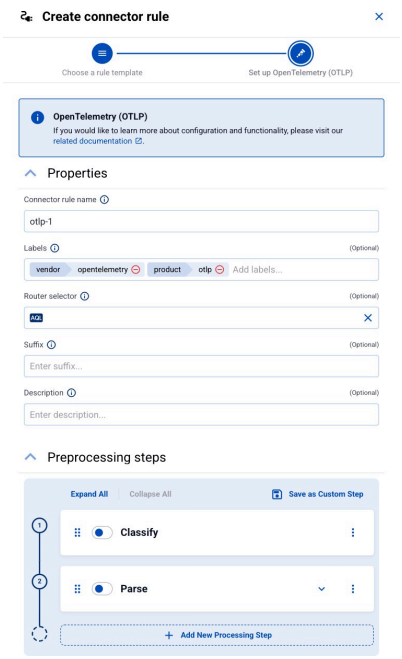
   To configure a different port, or to specify the protocol elements manually, select **Custom**.

   > **CAUTION:**
   >
   > TCP and UDP port range 29500–29519 is reserved for AxoRouter's internal listeners and cannot be used in connector rules.



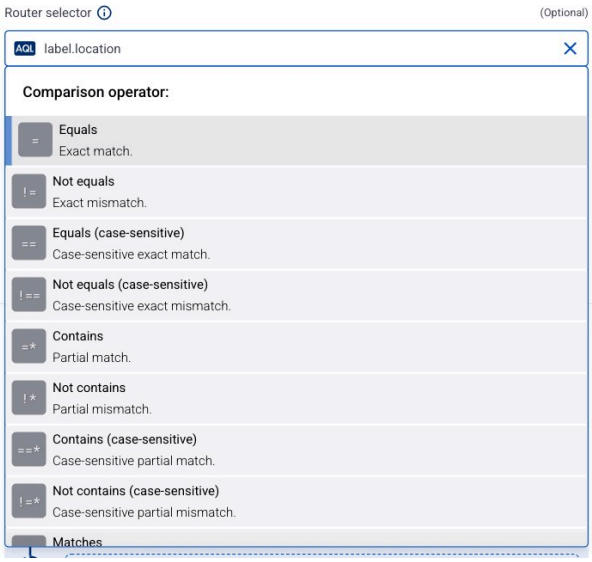
4. Configure the connector rule.

   1. Enter a name for the connector rule into the **Connector rule name** field.



2. (Optional) Add labels to the connector rule. You will be able to use these labels in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps. For details about the message schema and the related connector-specific fields, see the `meta.connector` object.

3. Set the **Router selector** for the connector rule. The selector determines which AxoRouter instances will have a connector based on this connector rule.



You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

- If you leave the router selector empty, it matches every AxoRouter instance.
- To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.

- If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

4. (Optional) Enter a **Suffix** for the connector rule. AxoConsole uses the suffix in the name of the connector instances that it creates on the AxoRouter hosts. The name of the connector instance follows the `<name-of-AxoRouter>-<suffix>` schema. For example, if the name of a matching AxoRouter instance is `axorouter-1`, and the suffix of the rule is `custom`, the name of the connector on that AxoRouter is `axorouter-1-custom`.

   If the **Suffix** field is empty, AxoConsole uses the name of the connector rule instead. For example, if the name of the connector rule is `otlp-1`, the name of the connector is `axorouter-1-otlp-1`.

5. (Optional) Enter a description for the rule.

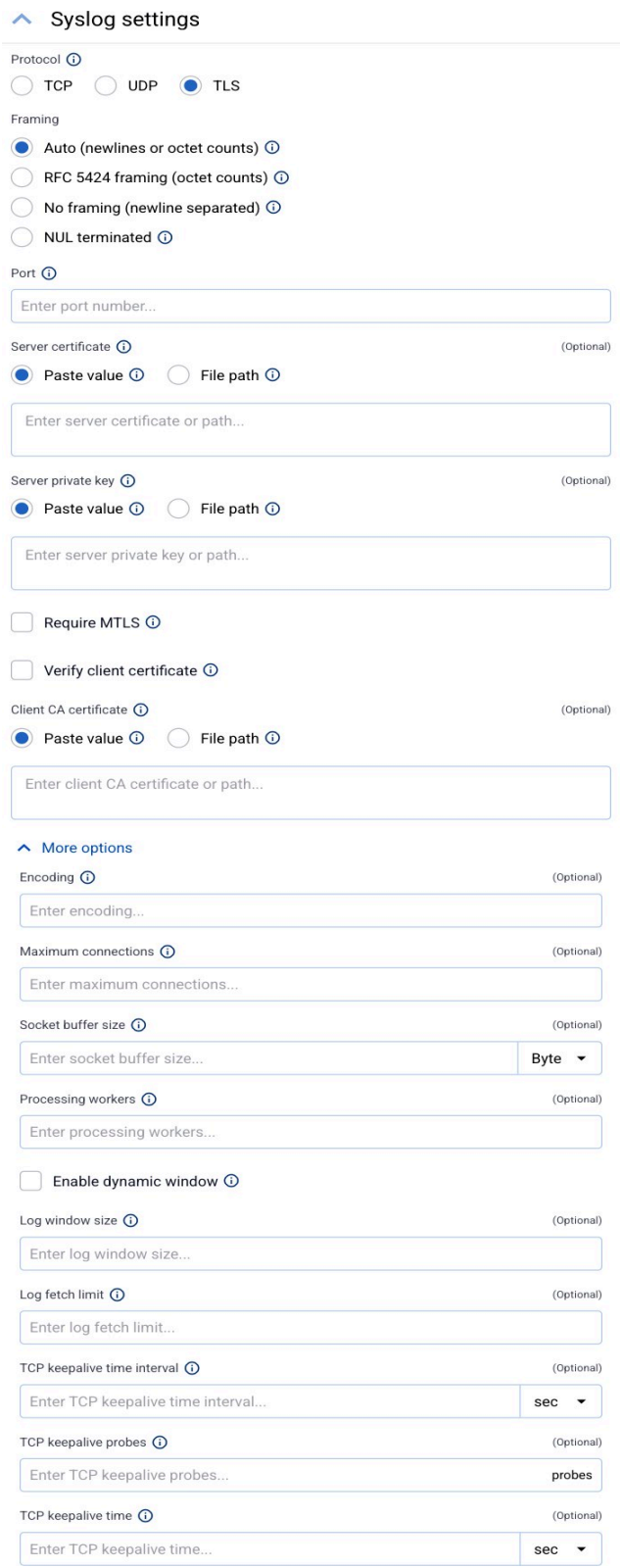
6. If needed, turn on the **Classify** and **Parse** steps in the **Preprocessing steps** section (they're off by default), so AxoRouter automatically identifies and parses messages sent by supported data sources. If your source is not listed, contact us.

   Enabling these options processes all data received by the connectors created based on this connector rule. If you want to apply classification and parsing more selectively, you can use the **Classify** and **Parse** processing steps in your **Flows**.

   Note that the **Parse** processing step requires **Classify** to be enabled. Parsing automatically parses the data from the content of the message, and replaces the message content (the `log.body` field in the internal message schema) with the structured information.

7. (Optional) If needed, you can add other preprocessing steps to the connector rule. You can use the same processing steps as in flows. These can be useful if you use a dedicated connector rule to fix or annotate data coming from some specific sources. To reuse the preprocessing steps in other connector rules or flows, select **Save as Custom Step**.

5. Select the protocol to use for receiving syslog data: **TCP**, **UDP**, or **TLS**.



When using TLS, set the paths for the certificates and keys used for the TLS-encrypted communication with the clients. For details, see Prerequisites.

> **Note**
>
> **These settings apply to every AxoRouter that matches the Router selector of the connector rule. If you paste values for the server certificate and server private key, every matching AxoRouter uses the same shared certificate, which is not recommended. Instead, use a separate certificate for each AxoRouter by setting the file path option, or create a separate connector rule for each AxoRouter.**

- **Server certificate**: The certificate that AxoRouter shows to the clients.
  - To provide the server certificate from the AxoConsole, select **Paste value** and paste the server certificate. AxoConsole will automatically deploy the server certificate to the related AxoRouters.
  - To provide the server certificate from a file, select **File path** and enter the path where the certificate file is located (for example, `/etc/axorouter/user-config/server-certificate.pem`). The path must be mounted into the container, `/etc/axorouter` is mounted by default. You must manually copy that file to its place on every affected AxoRouter, or you can use the configuration management system of your choice.
- **Server private key**: The private key of the server certificate.
  - To provide the server private key from the AxoConsole, select **Paste value** and paste the server private key. AxoConsole will automatically deploy the server private key to the related AxoRouters.
  - To provide the server private key from a file, select **File path** and enter the path where the private key file is located (for example, `/etc/axorouter/user-config/server-private-key.pem`). The path must be mounted into the container, `/etc/axorouter` is mounted by default. You must manually copy that file to its place on every affected AxoRouter, or you can use the configuration management system of your choice.
- **Client CA certificate**: The CA certificate that AxoRouter uses to verify the certificate of the sender if both **Verify client certificate** and **Require MTLS** are enabled.
  - To provide the CA certificate from the AxoConsole, select **Paste value** and paste the CA certificate. AxoConsole will automatically deploy the CA certificate to the related AxoRouters.
  - To provide the CA certificate from a file, select **File path** and enter the path where the certificate file is located (for example, `/etc/axorouter/user-config/ca-certificate.pem`). The path must be mounted into the container,

`/etc/axorouter` is mounted by default. You must manually copy that file to its place on every affected AxoRouter, or you can use the configuration management system of your choice.

Warning

When validating a certificate, the entire certificate chain must be valid, including the CA certificate. If any certificate of the chain is invalid, Axoflow will reject the connection.

Here's how the different combination of **Verify client certificate** and **Require MTLS** affects the verification of the client certificate and the connection.

- If **Require MTLS** is disabled, the client doesn't have to provide a certificate.
- If **Verify client certificate** is disabled, only the existence of the certificate is checked, but it does not have to be valid. AxoRouter accepts the certificate even if it is expired, signed by an unknown CA, or its CN and the name of the machine mismatches.

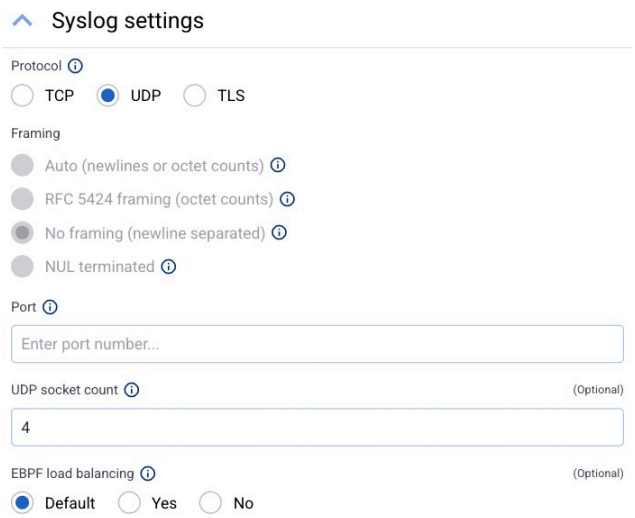
The following table summarizes the possible options and their results depending on the certificate of the peer.

		The client has:		
		No certificate	**Invalid certificate**	**Valid certificate**
AxoRouter settings	**Verify client certificate** disabled **Require MTLS** disabled	TLS encryption	TLS encryption	TLS encryption
	Verify client certificate enabled **Require MTLS** disabled	TLS encryption	Connection rejected	TLS encryption
	Verify client certificate disabled **Require MTLS** enabled	Connection rejected	TLS encryption	TLS encryption
	Verify client certificate enabled **Require MTLS** enabled	Connection rejected	Connection rejected	TLS encryption

6. (Optional) If explicitly needed for your use case, you can configure **Framing** manually when using the **TCP** and **TLS** protocols. Otherwise, leave it on **Auto**.
   - **Auto (newlines or octet counts)**: Auto-detect RFC 5424 framing or no framing. NUL terminated framing cannot be autodetected currently.
   - **RFC 5424 framing (octet counts)**: The payload contains the length of the message as specified in RFC6587 3.4.1.
   - **No framing (newline separated)**: Traditional newline-separated messages (non-transparent-framing) as specified in RFC6587 3.4.2.
   - **NUL terminated**: Proprietary protocol where (possibly multi-line) messages are separated by a NUL byte. Use this when receiving messages from Splunk Heavy Forwarders using the Axoflow Forwarder Add-on.
7. Set the **Port** of the connector. The port number must be unique on the AxoRouter host. AxoConsole will not provision a connector to an AxoRouter if it would cause a port collision, but other software on the given host may already be using the chosen port number (for example, an SSH server on TCP port 22). In this case, AxoRouter won't be able to reload the configuration and it will indicate an error.
8. (Optional) If you're expecting high-volume UDP traffic on your AxoRouter instances that will have this connector, enable UDP loadbalancing by entering the number of UDP sockets to use into the **UDP socket count** field. The maximum recommended value is the number of cores available in the AxoRouter host.

   If the value of **UDP socket count** is higher than 1 and **EBPF load balancing** is disabled, messages from a specific source host is always processed by the same socket (based on its IP address:port).

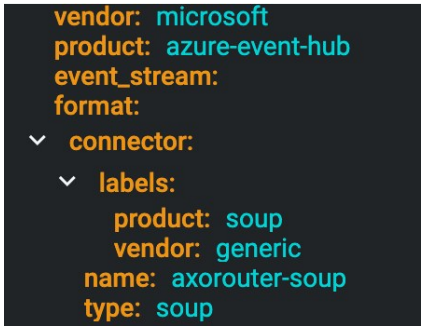
   To loadbalance every message, including messages coming from a single host (for example, a high-traffic firewall that send a high volume of messages from a single IP:port pair), you can enable the **EBPF load balancing** option. Note that if you enable EBPF loadbalancing, messages of the same high-traffic source may be processed out of order. For details on how eBPF loadbalancing works, see the Scaling syslog to 1M EPS with eBPF blog post.



9. (Optional) If needed for your environment, set protocol-specific connector options as needed.

   You can also modify the *product* and *vendor* labels of the connector. In that case, Axoflow will treat the incoming messages as it was received and classified as data from the specified product. This is useful if you want to send data from a specific product to a dedicated port.

   These labels and other parameters of the connector will be available under the `meta.connector` key as metadata for the messages received via the connector, and can be used in routing decisions and processing steps. You can check the metadata of the messages using log tapping.



10. Select **Add**.

    Axoflow automatically creates connectors on the AxoRouter hosts that match the **Router selector**.

    Make sure to enable the ports you've configured in the connector on the firewall of the AxoRouter host, and on other firewalls between the AxoRouter host and your data sources.

Protocol-specific connector options

- **Encoding**: The character set of the messages, for example, `UTF-8`.
- **Maximum connections**: The maximum number of simultaneous connections the connector can receive.
- **Socket buffer size**: The size of the socket buffer (in bytes).

  An undersized receive buffer is a common cause of UDP packet loss under load. To prevent this, raise the *socket buffer size* setting to the high-traffic recommended value of 32 MiB. The `net.core.rmem_max` `sysctl` setting is the upper limit of this value on the host operating system.
- **Processing workers**: Number of worker threads to use to process the messages received by the source connector. Note that this enables parallel processing, which might cause that the messages will be processed out of order. Increasing the number of worker threads can drastically improve the performance of the source. For details, see the AxoSyslog documentation.

Flow-control options

- **Log window size**: The size (number of messages) of the initial window used in flow control. For details, see the AxoSyslog documentation.

  If you change the value of **Log window size**, the change will affect only new connections, the **Log window size** of already established connections will not change. To apply the new **Log window size** value to every connection, restart the `axorouter.service` service on the **Services** page of the matching AxoRouters. A simple configuration reload is *NOT* sufficient.

  If the connector rule is receiving data using the UDP protocol, always restart the `axorouter.service` service on the **Services** page of the matching AxoRouters.
- **Enable dynamic window**: Enables the dynamic sizing of memory queues by sharing the configured **Log window size** between active incoming connections dynamically, instead of allocating the same window size to each possible connection. Only available for TCP and TLS.
- **Log fetch limit**: The maximum number of messages fetched from a source during a single poll loop. For details, see the AxoSyslog documentation.

TCP options

- **TCP keepalive time interval**: The interval (number of seconds) between subsequential keepalive probes, regardless of the traffic exchanged in the connection.
- **TCP keepalive probes**: The number of unacknowledged probes to send before considering the connection dead.
- **TCP keepalive time**: The interval (in seconds) between the last data packet sent and the first keepalive probe.

TLS options

For TLS, you can use the TCP-specific options, and also the following:

- **Require MTLS**: If enabled, the peers must have a TLS certificate, otherwise AxoRouter will reject the connection.
- **Verify client certificate**: If enabled, AxoRouter verifies certificate of the peer, and rejects connections with invalid certificates.

Labels

The AxoRouter syslog connector adds the following meta labels:

Analytics label	Message field	Value
`connector.type`	`meta.connector.type`	`syslog`
`connector.name`	`meta.connector.name`	The **Name** of the connector
`connector.port`	`meta.connector.port`	The port number where the connector receives data

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

7.7 Webhook

Receive JSON events over HTTP or HTTPS POST requests, and set message fields from the request path.

Webhook connectors of AxoRouter can be used to receive log events through HTTP(S) POST requests.

You can specify static and dynamic URLs to receive the data. AxoRouter automatically parses the JSON payload of the request, and adds it to the `log.body` field of the message as a JSON object. Other types of payload (including invalid JSON objects) is added to the `log.body` field as a string. Note that you can add further parsing to the body using processing steps in Flows, for example, using FilterX or Regex processing steps.

If your clients send several events in a single request (for example, line-separated records or concatenated JSON objects), use a Generic HTTP connector instead. Generic HTTP connectors split the body of the request into separate messages.

Prerequisites

To receive data via HTTPS, you'll need a key and a certificate that the connector will show to the clients.

If you want to enable TLS encryption for this connector to encrypt the communication with the sources, you'll need to set appropriate keys and certificates.

> **CAUTION:**
>
> Copy the keys and certificates to AxoRouter before starting to configure the connector. Otherwise, you won't be able to make configuration changes that require reloading the AxoRouter service, including starting log tapping or flow tapping.

Note the following points:

- Keys and certificates must be in PEM format.
- If the file contains a certificate chain, the file must begin with the certificate of the host, followed by the CA certificate that signed the certificate of the host, and any other signing CAs in order.
- If you manually copy these files to their place on the AxoRouter host, make the files readable by the `axorouter` service.
  - The recommended path for certificates is under `/etc/axorouter/user-config/` (for example, `/etc/axorouter/user-config/tls-key.pem`). (If you need to use a different path, you have to append an option like `-v /your/path:/your/path` to the

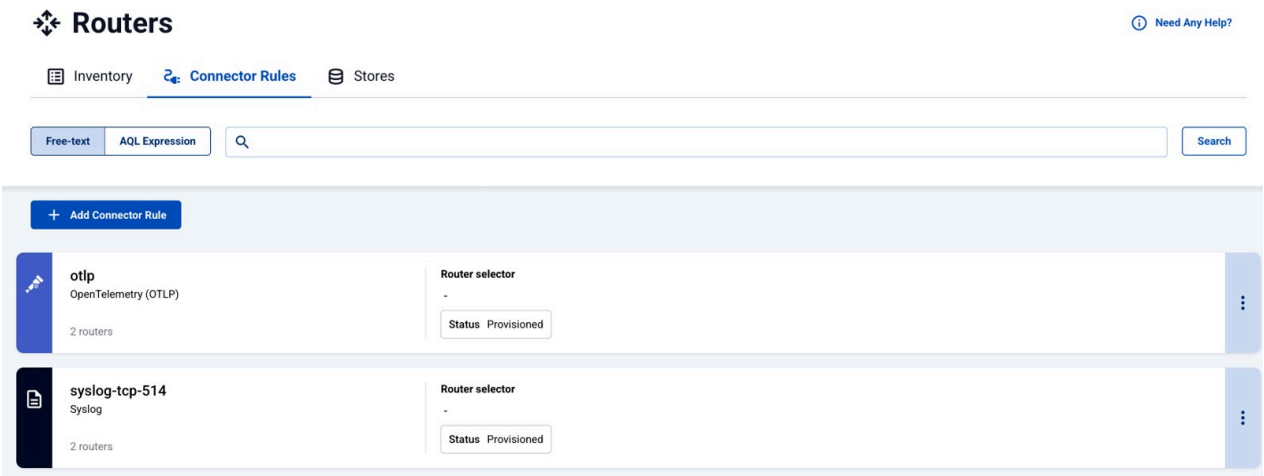
`AXOROUTER_PODMAN_ARGS` variable of `/etc/axorouter/container.env`.)

- When referring to the key or certificate when configuring the connector, use absolute paths (for example, `/etc/axorouter/user-config/tls-key.pem`).

Add new webhook connector

To add a new connector to an AxoRouter host, complete the following steps:

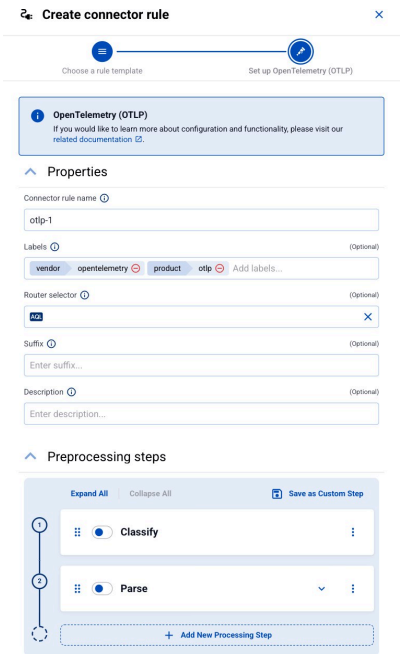
1. Select **Routers > Connector Rules > Add Connector Rule**. (Alternatively, you can select **Add Connector > Add a connector rule** on the **Connectors** page of an AxoRouter host.)



2. Select **Webhook**.

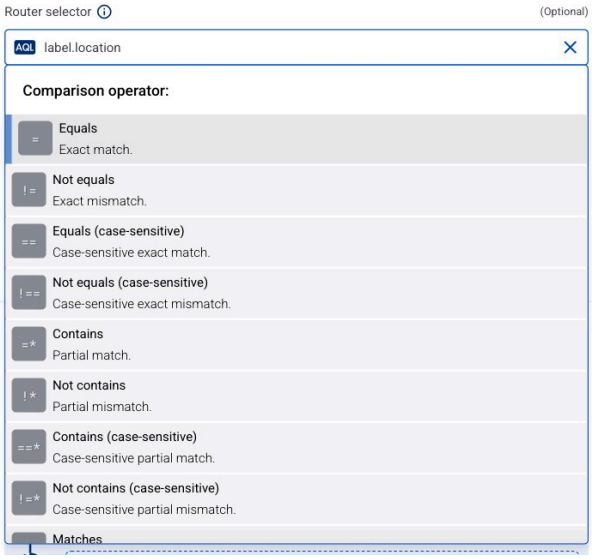
3. Configure the connector rule.

    1. Enter a name for the connector rule into the **Connector rule name** field.



2. (Optional) Add labels to the connector rule. You will be able to use these labels in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps. For details about the message schema and the related connector-specific fields, see the `meta.connector` object.

3. Set the **Router selector** for the connector rule. The selector determines which AxoRouter instances will have a connector based on this connector rule.



   You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

   - If you leave the router selector empty, it matches every AxoRouter instance.
   - To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.
   - If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

4. (Optional) Enter a **Suffix** for the connector rule. AxoConsole uses the suffix in the name of the connector instances that it creates on the AxoRouter hosts. The name of the connector instance follows the `<name-of-AxoRouter>-<suffix>` schema. For example, if the name of a matching AxoRouter instance is `axorouter-1`, and the suffix of the rule is `custom`, the name of the connector on that AxoRouter is `axorouter-1-custom`.

   If the **Suffix** field is empty, AxoConsole uses the name of the connector rule instead. For example, if the name of the connector rule is `otlp-1`, the name of the connector is `axorouter-1-otlp-1`.

5. (Optional) Enter a description for the rule.

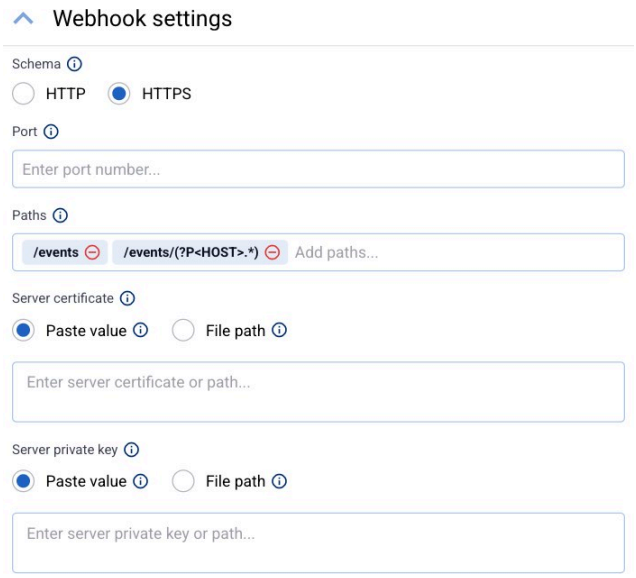
6. If needed, turn on the **Classify** and **Parse** steps in the **Preprocessing steps** section (they're off by default), so AxoRouter automatically identifies and parses messages sent by supported data sources. If your source is not listed, contact us.

   Enabling these options processes all data received by the connectors created based on this connector rule. If you want to apply classification and parsing more selectively, you can use the **Classify** and **Parse** processing steps in your **Flows**.

   Note that the **Parse** processing step requires **Classify** to be enabled. Parsing automatically parses the data from the content of the message, and replaces the message content (the `log.body` field in the internal message schema) with the structured information.

7. (Optional) If needed, you can add other preprocessing steps to the connector rule. You can use the same processing steps as in flows. These can be useful if you use a dedicated connector rule to fix or annotate data coming from some specific sources. To reuse the preprocessing steps in other connector rules or flows, select **Save as Custom Step**.

4. In the **Schema** field, select the protocol you want to use: **HTTP** or **HTTPS**.



5. Set the port number where the webhook will receive the POST requests, for example, `8080`.

   CAUTION:

   TCP and UDP port range 29500–29519 is reserved for AxoRouter's internal listeners and cannot be used in connector rules.

6. Set the endpoints where the webhook will receive data in the **Paths** field. You can use static paths, or regular expressions. In regular expressions you can use named capture groups to automatically set macro values in AxoRouter. For example, the

`/events/(?P<HOST>.*)` path sets the hostname for the data received in the request based on the second part of the URL: a request to the `/events/my-example-host` URL sets the host field of that message to `my-example-host`.

By default, the `/events` and `/events/(?P<HOST>.*)` paths are active.

7. For HTTPS endpoints, set the path to the **Server certificate** and the **Server private key** options. AxoRouter uses these to encrypt the TLS channel.

> **Note**
>
> **These settings apply to every AxoRouter that matches the Router selector of the connector rule. If you paste values for the server certificate and server private key, every matching AxoRouter uses the same shared certificate, which is not recommended. Instead, use a separate certificate for each AxoRouter by setting the file path option, or create a separate connector rule for each AxoRouter.**

   - **Server certificate**: The certificate that AxoRouter shows to the clients.
     - To provide the server certificate from the AxoConsole, select **Paste value** and paste the server certificate. AxoConsole will automatically deploy the server certificate to the related AxoRouters.
     - To provide the server certificate from a file, select **File path** and enter the path where the certificate file is located (for example, `/etc/axorouter/user-config/server-certificate.pem`). The path must be mounted into the container, `/etc/axorouter` is mounted by default. You must manually copy that file to its place on every affected AxoRouter, or you can use the configuration management system of your choice.
   - **Server private key**: The private key of the server certificate.
     - To provide the server private key from the AxoConsole, select **Paste value** and paste the server private key. AxoConsole will automatically deploy the server private key to the related AxoRouters.
     - To provide the server private key from a file, select **File path** and enter the path where the private key file is located (for example, `/etc/axorouter/user-config/server-private-key.pem`). The path must be mounted into the container, `/etc/axorouter` is mounted by default. You must manually copy that file to its place on every affected AxoRouter, or you can use the configuration management system of your choice.
8. Select **Add**.

Axoflow automatically creates connectors on the AxoRouter hosts that match the **Router selector**.

Make sure to enable the ports you've configured in the connector on the firewall of the AxoRouter host, and on other firewalls between the AxoRouter host and your data sources.

9. Send a test request to the webhook.

    1. Open the **Overview** tab of the AxoRouter host where you've created the webhook connector and check the IP address or FQDN of the host.

    2. Open a terminal on your computer and send a POST request to the path you've configured in the webhook.

    ```
    curl -X POST -H 'Content-Type: application/json' --data '{"host":"myhost",
    "body": "sample webhook message" }' <router-IP-address>:<webhook-port-
    number>/<webhook-path>/
    ```

    For example, if you've used the default values of the webhook connector you can use:

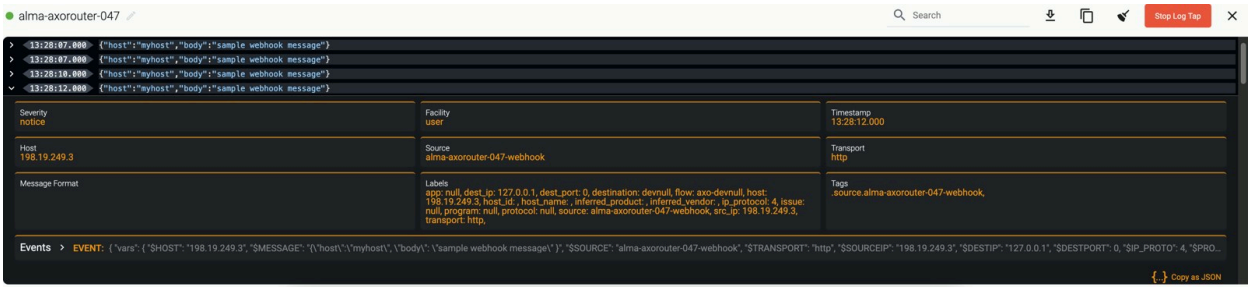
    ```
    curl -X POST -H 'Content-Type: application/json' --data '{"host":"myhost",
    "body": "sample webhook message" }' <router-IP-address>/events/
    ```

    Expected output:

    ```
    {"status": "received"}
    ```

Note

You can also use Log tapping on the AxoRouter host to see the incoming messages.



Metadata fields

The AxoRouter webhook connector adds the following fields to the `meta` variable:

Analytics label	Message field	Value
`connector.type`	`meta.connector.type`	`webhook`
`connector.name`	`meta.connector.name`	The **Name** of the connector
`connector.port`	`meta.connector.port`	The port number where the connector receives data
`connector.labels.product`	`meta.connector.labels.product`	Default value: `webhook`
`connector.labels.vendor`	`meta.connector.labels.vendor`	Default value: `generic`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

7.8 Generic HTTP

Receive batches of log events that clients post over HTTP or HTTPS, and choose how AxoRouter splits the request body into messages.

Generic HTTP connectors of AxoRouter can receive log events from any client that posts them over HTTP or HTTPS. Use this connector for senders that post logs over HTTP. If Axoflow has a vendor-specific solution to ingest data, for example the Splunk HEC connector, use that instead.

AxoRouter doesn't parse the payload of the request: it splits the request body into messages according to the **Message mode** you select, and adds each message to the `log.body` field as a string. You can add parsing to the body using processing steps in Flows, for example, using FilterX or Regex processing steps.

If every request of your clients carries a single JSON event, consider using a Webhook connector instead. Webhook connectors parse the JSON payload automatically, and can set message fields from the path of the request.

Prerequisites

To receive data over HTTPS, you'll need a key and a certificate that the connector shows to the clients.

If you want to enable TLS encryption for this connector to encrypt the communication with the sources, you'll need to set appropriate keys and certificates.

> **CAUTION:**
>
> Copy the keys and certificates to AxoRouter before starting to configure the connector. Otherwise, you won't be able to make configuration changes that require reloading the AxoRouter service, including starting log tapping or flow tapping.

Note the following points:

- Keys and certificates must be in PEM format.
- If the file contains a certificate chain, the file must begin with the certificate of the host, followed by the CA certificate that signed the certificate of the host, and any other signing CAs in order.
- If you manually copy these files to their place on the AxoRouter host, make the files readable by the `axorouter` service.
  - The recommended path for certificates is under `/etc/axorouter/user-config/` (for example, `/etc/axorouter/user-config/tls-key.pem`). (If you need to use a different path, you have to append an option like `-v /your/path:/your/path` to the

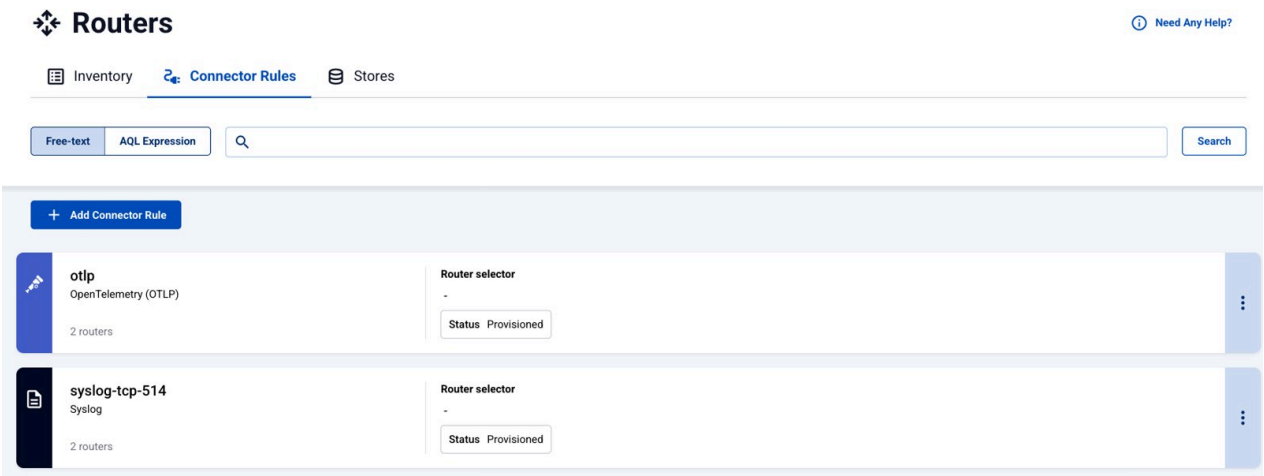
`AXOROUTER_PODMAN_ARGS` variable of `/etc/axorouter/container.env`.)

- When referring to the key or certificate when configuring the connector, use absolute paths (for example, `/etc/axorouter/user-config/tls-key.pem`).

Add new Generic HTTP connector

To add a new connector to an AxoRouter host, complete the following steps:

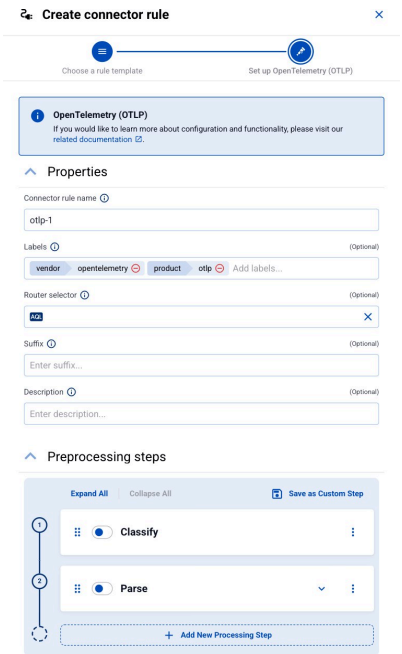
1. Select **Routers > Connector Rules > Add Connector Rule**. (Alternatively, you can select **Add Connector > Add a connector rule** on the **Connectors** page of an AxoRouter host.)



2. Select **Generic HTTP**.

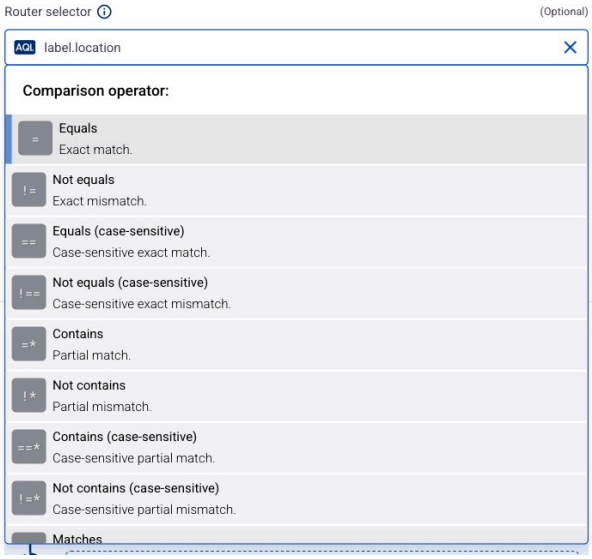
3. Configure the connector rule.

    1. Enter a name for the connector rule into the **Connector rule name** field.



2. (Optional) Add labels to the connector rule. You will be able to use these labels in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps. For details about the message schema and the related connector-specific fields, see the `meta.connector` object.

3. Set the **Router selector** for the connector rule. The selector determines which AxoRouter instances will have a connector based on this connector rule.



   You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

   - If you leave the router selector empty, it matches every AxoRouter instance.
   - To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.
   - If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

4. (Optional) Enter a **Suffix** for the connector rule. AxoConsole uses the suffix in the name of the connector instances that it creates on the AxoRouter hosts. The name of the connector instance follows the `<name-of-AxoRouter>-<suffix>` schema. For example, if the name of a matching AxoRouter instance is `axorouter-1`, and the suffix of the rule is `custom`, the name of the connector on that AxoRouter is `axorouter-1-custom`.

   If the **Suffix** field is empty, AxoConsole uses the name of the connector rule instead. For example, if the name of the connector rule is `otlp-1`, the name of the connector is `axorouter-1-otlp-1`.

5. (Optional) Enter a description for the rule.

6. If needed, turn on the **Classify** and **Parse** steps in the **Preprocessing steps** section (they're off by default), so AxoRouter automatically identifies and parses messages sent by supported data sources. If your source is not listed, contact us.

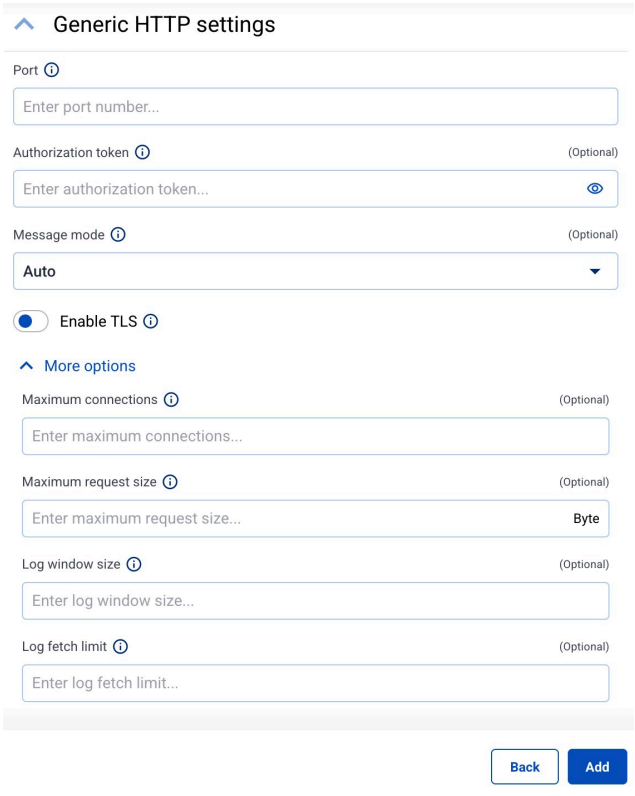
   Enabling these options processes all data received by the connectors created based on this connector rule. If you want to apply classification and parsing more selectively, you can use the **Classify** and **Parse** processing steps in your **Flows**.

   Note that the **Parse** processing step requires **Classify** to be enabled. Parsing automatically parses the data from the content of the message, and replaces the message content (the `log.body` field in the internal message schema) with the structured information.

7. (Optional) If needed, you can add other preprocessing steps to the connector rule. You can use the same processing steps as in flows. These can be useful if you use a dedicated connector rule to fix or annotate data coming from some specific sources. To reuse the preprocessing steps in other connector rules or flows, select **Save as Custom Step**.

4. Set the **Port** number where the connector receives the requests, for example, `8080`.

> **CAUTION:**
>
> TCP and UDP port range 29500–29519 is reserved for AxoRouter's internal listeners and cannot be used in connector rules.



5. (Optional) Enter an **Authorization token**. If you set it, the clients must send this token in the `Authorization` header of every request, and AxoRouter refuses the requests that don't carry it. The token is compared verbatim against the whole Authorization header value, so for example if a client sends `Authorization: Bearer xyz`, the value of **Authorization token** must be `Bearer xyz`.

6. Select the **Message mode** that matches how your clients send the data. It determines how AxoRouter splits the body of a request into messages.

    - **Auto**: Detects a single JSON document, concatenated JSON objects, or line-separated records. This is the default value.
    - **JSON**: The body is a single JSON document, or a series of concatenated JSON objects.
    - **Line separated**: Every line of the body is a separate message.
    - **Single**: The whole body is a single message.

7. (Optional) To receive data over HTTPS, select **Enable TLS**, then set the certificate and the private key that AxoRouter shows to the clients.

> ### Note
>
> **These settings apply to every AxoRouter that matches the Router selector of the connector rule. If you paste values for the server certificate and server private key, every matching AxoRouter uses the same shared certificate, which is not recommended. Instead, use a separate certificate for each AxoRouter by setting the file path option, or create a separate connector rule for each AxoRouter.**

    - **Server certificate**: The certificate that AxoRouter shows to the clients.
        - To provide the server certificate from the AxoConsole, select **Paste value** and paste the server certificate. AxoConsole will automatically deploy the server certificate to the related AxoRouters.
        - To provide the server certificate from a file, select **File path** and enter the path where the certificate file is located (for example, `/etc/axorouter/user-config/server-certificate.pem`). The path must be mounted into the container, `/etc/axorouter` is mounted by default. You must manually copy that file to its place on every affected AxoRouter, or you can use the configuration management system of your choice.
    - **Server private key**: The private key of the server certificate.
        - To provide the server private key from the AxoConsole, select **Paste value** and paste the server private key. AxoConsole will automatically deploy the server private key to the related AxoRouters.

- To provide the server private key from a file, select **File path** and enter the path where the private key file is located (for example, `/etc/axorouter/user-config/server-private-key.pem`). The path must be mounted into the container, `/etc/axorouter` is mounted by default. You must manually copy that file to its place on every affected AxoRouter, or you can use the configuration management system of your choice.

8. (Optional) Select **More options** to change the advanced options of the connector.
9. Select **Add**.

   Axoflow automatically creates connectors on the AxoRouter hosts that match the **Router selector**.

   Make sure to enable the ports you've configured in the connector on the firewall of the AxoRouter host, and on other firewalls between the AxoRouter host and your data sources.
10. Send a test request to the connector.

    1. Open the **Overview** tab of the AxoRouter host where you've created the connector and check the IP address or FQDN of the host.
    2. Open a terminal on your computer and send a POST request to the port you've configured in the connector.

    ```
    curl -X POST -H 'Content-Type: application/json' --data '{"message": "sample generic http message"}' http://<router-IP-address>:<connector-port-number>/
    ```

    If you've set an **Authorization token**, send it in the `Authorization` header as well:

    ```
    curl -X POST -H 'Content-Type: application/json' -H 'Authorization: <your-token>' --data '{"message": "sample generic http message"}' http://<router-IP-address>:<connector-port-number>/
    ```

> ## Note
>
> **You can also use Log tapping on the AxoRouter host to see the incoming messages.**

Advanced options

Select **More options** on the connector rule to set the following options.

- **Maximum connections**: The maximum number of connections that the connector accepts at the same time.
- **Maximum request size**: The maximum accepted size of a request body, in bytes. AxoRouter refuses the requests that are larger than this.

You can also adjust the flow-control related options.

- **Log window size**: The size (number of messages) of the initial window used in flow control. For details, see the AxoSyslog documentation.

  If you change the value of **Log window size**, the change will affect only new connections, the **Log window size** of already established connections will not change. To apply the new **Log window size** value to every connection, restart the `axorouter.service` service on the **Services** page of the matching AxoRouters. A simple configuration reload is *NOT* sufficient.

  If the connector rule is receiving data using the UDP protocol, always restart the `axorouter.service` service on the **Services** page of the matching AxoRouters.

- **Log fetch limit**: The maximum number of messages fetched from a source during a single poll loop. For details, see the AxoSyslog documentation.

Metadata fields

The AxoRouter Generic HTTP connector adds the following fields to the `meta` variable:

Analytics label	Message field	Value
`connector.type`	`meta.connector.type`	`genericHttp`
`connector.name`	`meta.connector.name`	The **Name** of the connector
`connector.port`	`meta.connector.port`	The port number where the connector receives data
`connector.labels.product`	`meta.connector.labels.product`	Default value: `generic-http`
`connector.labels.vendor`	`meta.connector.labels.vendor`	Default value: `generic`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

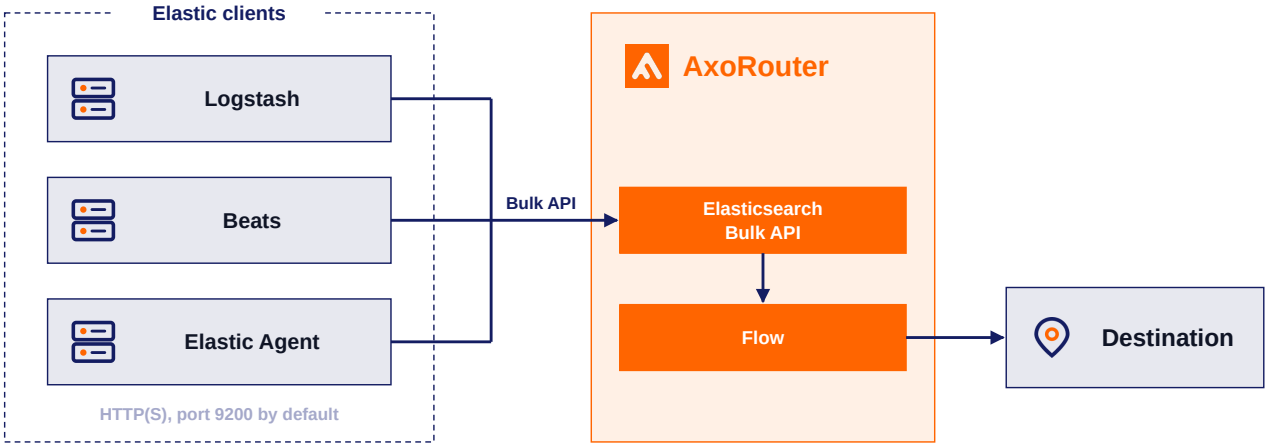
- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

7.9 Elasticsearch Bulk API

Receive data from Beats, Logstash, Elastic Agent, and other clients that post to the Elasticsearch Bulk API.

Elasticsearch Bulk API connectors of AxoRouter can receive log events from Beats, Logstash, Elastic Agent, and any other client that posts to the Elasticsearch Bulk API. Point such a client at AxoRouter instead of your Elasticsearch cluster, and Axoflow ingests the documents it sends.

> **Note**
>
> **This connector receives data that a client pushes to AxoRouter. To collect data that's already stored in an Elasticsearch index, use the Elasticsearch source instead, which polls the Elasticsearch `_search` API.**



Prerequisites

To receive data over HTTPS, you'll need a key and a certificate that the connector shows to the clients.

If you want to enable TLS encryption for this connector to encrypt the communication with the sources, you'll need to set appropriate keys and certificates.

> **CAUTION:**
>
> Copy the keys and certificates to AxoRouter before starting to configure the connector. Otherwise, you won't be able to make configuration changes that require reloading the AxoRouter service, including starting log tapping or flow tapping.

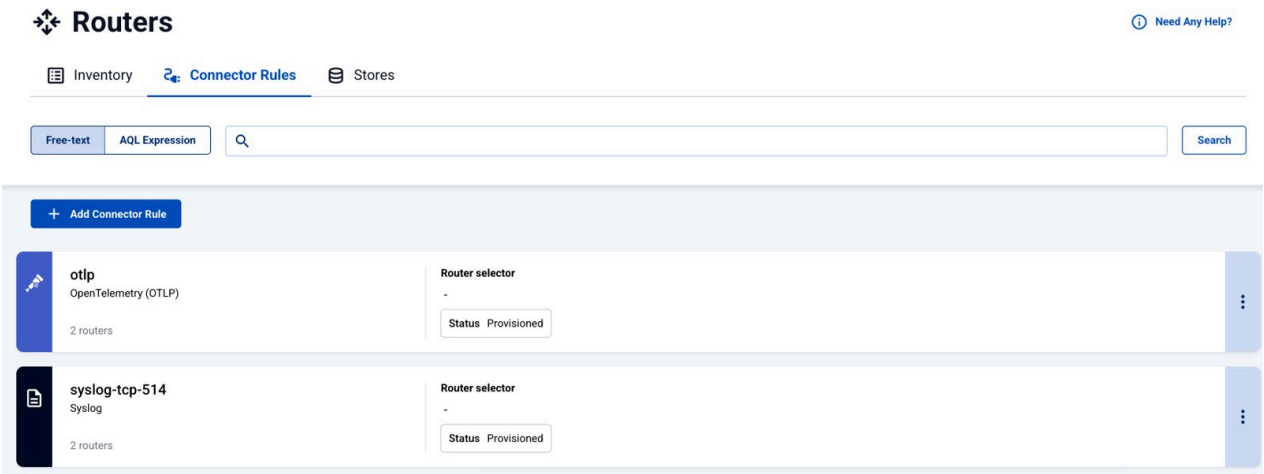
Note the following points:

- Keys and certificates must be in PEM format.
- If the file contains a certificate chain, the file must begin with the certificate of the host, followed by the CA certificate that signed the certificate of the host, and any other signing CAs in order.
- If you manually copy these files to their place on the AxoRouter host, make the files readable by the `axorouter` service.
  - The recommended path for certificates is under `/etc/axorouter/user-config/` (for example, `/etc/axorouter/user-config/tls-key.pem`). (If you need to use a different path, you have to append an option like `-v /your/path:/your/path` to the `AXOROUTER_PODMAN_ARGS` variable of `/etc/axorouter/container.env`.)
  - When referring to the key or certificate when configuring the connector, use absolute paths (for example, `/etc/axorouter/user-config/tls-key.pem`).

Add new Elasticsearch Bulk API connector

To add a new connector to an AxoRouter host, complete the following steps:

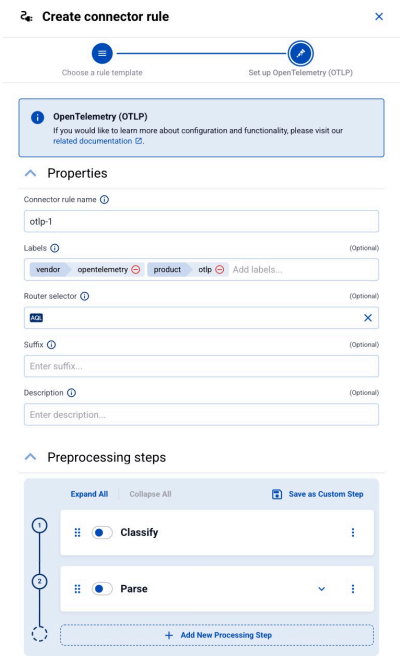
1. Select **Routers > Connector Rules > Add Connector Rule**. (Alternatively, you can select **Add Connector > Add a connector rule** on the **Connectors** page of an AxoRouter host.)



2. Select **Elasticsearch Bulk API**.

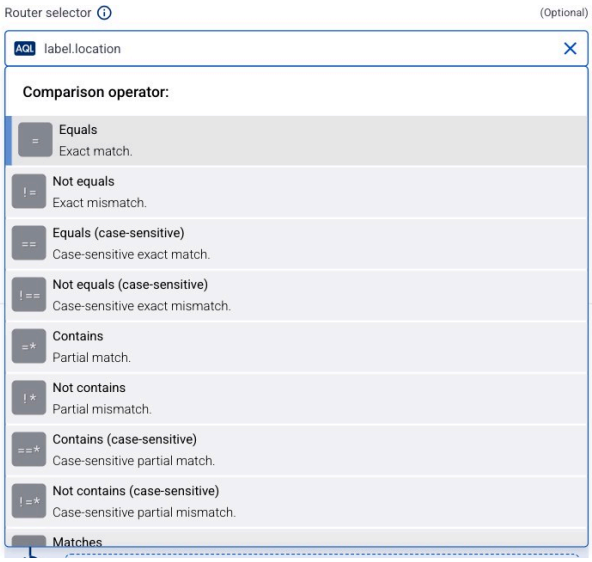
3. Configure the connector rule.

   1. Enter a name for the connector rule into the **Connector rule name** field.



2. (Optional) Add labels to the connector rule. You will be able to use these labels in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps. For details about the message schema and the related connector-specific fields, see the `meta.connector` object.

3. Set the **Router selector** for the connector rule. The selector determines which AxoRouter instances will have a connector based on this connector rule.



You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

- If you leave the router selector empty, it matches every AxoRouter instance.
- To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.

- If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

4. (Optional) Enter a **Suffix** for the connector rule. AxoConsole uses the suffix in the name of the connector instances that it creates on the AxoRouter hosts. The name of the connector instance follows the `<name-of-AxoRouter>-<suffix>` schema. For example, if the name of a matching AxoRouter instance is `axorouter-1`, and the suffix of the rule is `custom`, the name of the connector on that AxoRouter is `axorouter-1-custom`.

   If the **Suffix** field is empty, AxoConsole uses the name of the connector rule instead. For example, if the name of the connector rule is `otlp-1`, the name of the connector is `axorouter-1-otlp-1`.

5. (Optional) Enter a description for the rule.

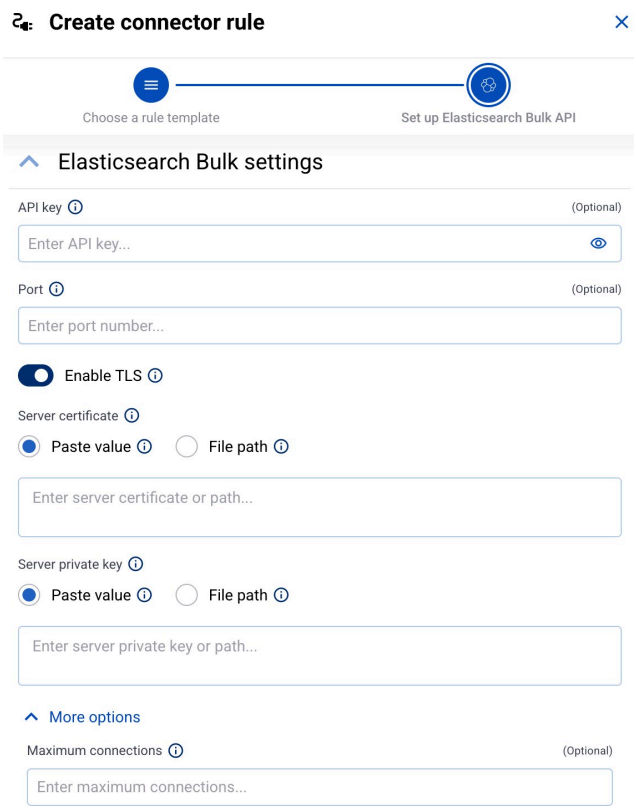
6. If needed, turn on the **Classify** and **Parse** steps in the **Preprocessing steps** section (they're off by default), so AxoRouter automatically identifies and parses messages sent by supported data sources. If your source is not listed, contact us.

   Enabling these options processes all data received by the connectors created based on this connector rule. If you want to apply classification and parsing more selectively, you can use the **Classify** and **Parse** processing steps in your **Flows**.

   Note that the **Parse** processing step requires **Classify** to be enabled. Parsing automatically parses the data from the content of the message, and replaces the message content (the `log.body` field in the internal message schema) with the structured information.

7. (Optional) If needed, you can add other preprocessing steps to the connector rule. You can use the same processing steps as in flows. These can be useful if you use a dedicated connector rule to fix or annotate data coming from some specific sources. To reuse the preprocessing steps in other connector rules or flows, select **Save as Custom Step**.

4. (Optional) Enter an **API key**. If you set it, the clients must authenticate with this Elasticsearch API key, and AxoRouter refuses the requests that don't carry it. Enter the key in its `id:api_key` form, the same form that the `api_key` option of Beats and Logstash uses, not the base64-encoded value. AxoRouter expects the clients to send it as `Authorization: ApiKey <base64-encoded id:api_key>`.



5. (Optional) Set the **Port** number where the connector receives the requests. If you leave it empty, the connector listens on port `9200` by default.

   > **CAUTION:**
   >
   > TCP and UDP port range 29500–29519 is reserved for AxoRouter's internal listeners and cannot be used in connector rules.

6. (Optional) To receive data over HTTPS, select **Enable TLS**, then set the certificate and the private key that AxoRouter shows to the clients.

   > ## Note
   >
   > **These settings apply to every AxoRouter that matches the Router selector of the connector rule. If you paste values for the server certificate and server private key, every matching AxoRouter uses the same shared certificate, which is not recommended. Instead, use a separate certificate for each AxoRouter by setting the file path option, or create a separate connector rule for each AxoRouter.**

   - **Server certificate**: The certificate that AxoRouter shows to the clients.
     - To provide the server certificate from the AxoConsole, select **Paste value** and paste the server certificate. AxoConsole will automatically deploy the server certificate to the related AxoRouters.

- To provide the server certificate from a file, select **File path** and enter the path where the certificate file is located (for example, `/etc/axorouter/user-config/server-certificate.pem`). The path must be mounted into the container, `/etc/axorouter` is mounted by default. You must manually copy that file to its place on every affected AxoRouter, or you can use the configuration management system of your choice.

- **Server private key**: The private key of the server certificate.
  - To provide the server private key from the AxoConsole, select **Paste value** and paste the server private key. AxoConsole will automatically deploy the server private key to the related AxoRouters.
  - To provide the server private key from a file, select **File path** and enter the path where the private key file is located (for example, `/etc/axorouter/user-config/server-private-key.pem`). The path must be mounted into the container, `/etc/axorouter` is mounted by default. You must manually copy that file to its place on every affected AxoRouter, or you can use the configuration management system of your choice.

7. (Optional) Select **More options** to change the advanced options of the connector.

8. Select **Add**.

   Axoflow automatically creates connectors on the AxoRouter hosts that match the **Router selector**.

   Make sure to enable the ports you've configured in the connector on the firewall of the AxoRouter host, and on other firewalls between the AxoRouter host and your data sources.

9. Send a test request to the connector.

   1. Open the **Overview** tab of the AxoRouter host where you've created the connector and check the IP address or FQDN of the host.

   2. Open a terminal on your computer and send a bulk request to the port you've configured in the connector. A bulk request is newline-delimited JSON, where an action line precedes every document, and the body ends with a newline.

```
curl -X POST -H "Authorization: ApiKey $(printf '%s' '<id>:<api-key>' |
base64 -w0)" -H 'Content-Type: application/x-ndjson' --data-binary
$'{"index":{"_index":"logs"}}\n{"@timestamp":"2026-09-
21T10:00:00.000Z","message":"sample bulk message"}\n' http://<router-IP-
address>:<connector-port-number>/_bulk
```

10. Configure your clients to send their data to AxoRouter. For example, in Beats or Logstash, set the Elasticsearch output to the address and the port of the connector.

> **Note**
>
> **You can also use Log tapping on the AxoRouter host to see the incoming messages.**

Advanced options

Select **More options** on the connector rule to set the following options.

- **Maximum connections**: The maximum number of connections that the connector accepts at the same time.
- **Maximum request size**: The maximum accepted size of a request body, in bytes. AxoRouter refuses the requests that are larger than this.

You can also adjust the flow-control related options.

- **Log window size**: The size (number of messages) of the initial window used in flow control. For details, see the AxoSyslog documentation.

  If you change the value of **Log window size**, the change will affect only new connections, the **Log window size** of already established connections will not change. To apply the new **Log window size** value to every connection, restart the `axorouter.service` service on the **Services** page of the matching AxoRouters. A simple configuration reload is *NOT* sufficient.

  If the connector rule is receiving data using the UDP protocol, always restart the `axorouter.service` service on the **Services** page of the matching AxoRouters.

- **Log fetch limit**: The maximum number of messages fetched from a source during a single poll loop. For details, see the AxoSyslog documentation.

Metadata fields

AxoRouter doesn't send the documents to an Elasticsearch cluster: it turns every document of the bulk request into a separate message, and adds the following fields to the `meta` variable:

Analytics label	Message field	Value
`connector.type`	`meta.connector.type`	`elasticsearch`

`connector.name`	`meta.connector.name`	The **Name** of the connector
`connector.port`	`meta.connector.port`	The port number where the connector receives data
`connector.labels.product`	`meta.connector.labels.product`	Default value: `elasticsearch-bulk`
`connector.labels.vendor`	`meta.connector.labels.vendor`	Default value: `elastic`
`envelope.elasticsearch.operation`	`meta.envelope.elasticsearch.operation`	The operation of the Bulk API action line that accompanies the document, for example, `index` or `create`
`envelope.elasticsearch.params`	`meta.envelope.elasticsearch.params`	The parameters of the action line, for example, `_index` and `_id`

Depending on the content of the document, AxoRouter sets the following fields as well:

- If the document has an `@timestamp` field, AxoRouter uses its value as `log.time_unix_nano`, the timestamp of the message. Both the ISO 8601 format and epoch milliseconds are accepted.
- If the document has an `agent.type` field (Beats and Elastic Agent set it on every document), AxoRouter also reads the following fields:
  - `host.name` becomes `meta.host.name`.
  - `event.dataset`, or `agent.type` if the document has no dataset, becomes `meta.service.name`.
  - If the document has a `message` field, its value becomes `log.body`, and the rest of the document becomes `log.attributes`.
- In every other case the document is added to the `log.body` field as a string. You can add further parsing to the body using processing steps in Flows, for example, using FilterX or Regex processing steps.

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

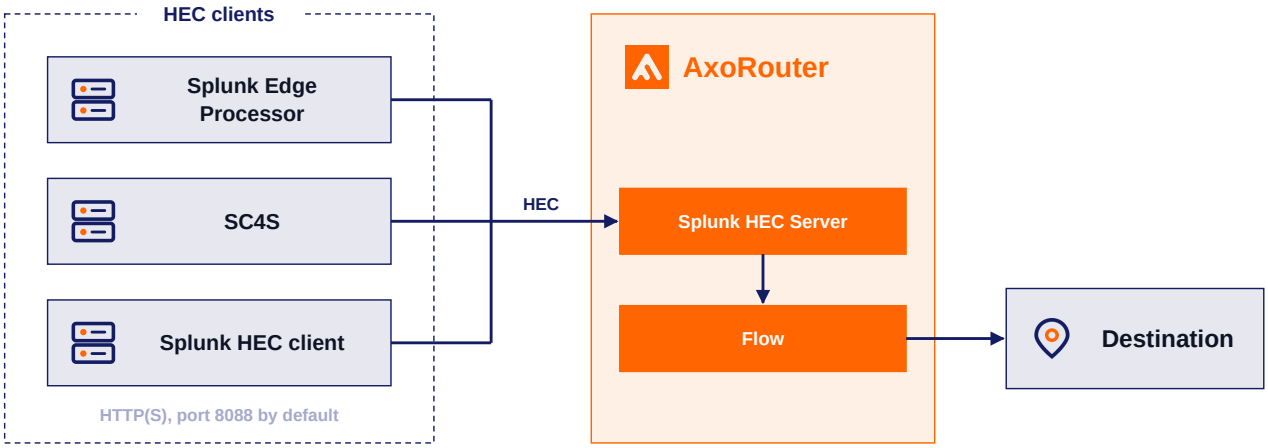
- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

7.10 Splunk HEC Server

Receive data from Splunk forwarders and other clients that post to the Splunk HTTP Event Collector (HEC) API.

Splunk HEC Server connectors of AxoRouter can receive log events from Splunk forwarders and any other client that posts to the Splunk HTTP Event Collector (HEC), for example, Splunk Edge Processor, Splunk OpenTelemetry Collector, or Splunk Connect for Syslog (SC4S). Point such a client at AxoRouter instead of your Splunk HEC endpoint, and Axoflow ingests the events it sends.

> **Note**
>
> **To collect data from a Splunk Heavy Forwarder with the Axoflow Forwarder for Splunk App, see Heavy Forwarder instead.**



Prerequisites

To receive data over HTTPS, you'll need a key and a certificate that the connector shows to the clients.

If you want to enable TLS encryption for this connector to encrypt the communication with the sources, you'll need to set appropriate keys and certificates.

> **CAUTION:**
>
> Copy the keys and certificates to AxoRouter before starting to configure the connector. Otherwise, you won't be able to make configuration changes that require reloading the AxoRouter service, including starting log tapping or flow tapping.

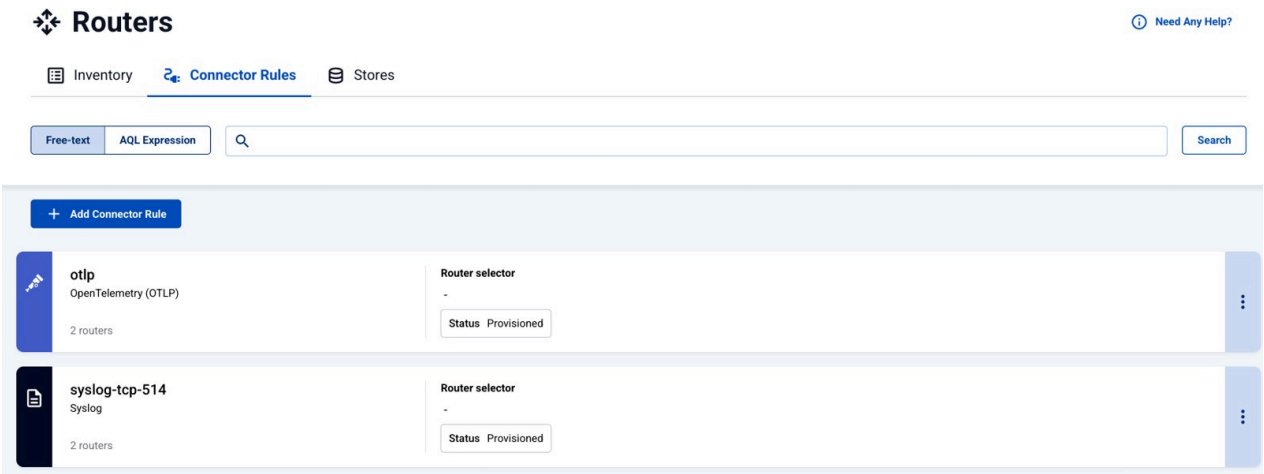
Note the following points:

- Keys and certificates must be in PEM format.
- If the file contains a certificate chain, the file must begin with the certificate of the host, followed by the CA certificate that signed the certificate of the host, and any other signing CAs in order.
- If you manually copy these files to their place on the AxoRouter host, make the files readable by the `axorouter` service.
  - The recommended path for certificates is under `/etc/axorouter/user-config/` (for example, `/etc/axorouter/user-config/tls-key.pem`). (If you need to use a different path, you have to append an option like `-v /your/path:/your/path` to the `AXOROUTER_PODMAN_ARGS` variable of `/etc/axorouter/container.env`.)
  - When referring to the key or certificate when configuring the connector, use absolute paths (for example, `/etc/axorouter/user-config/tls-key.pem`).

Add new Splunk HEC Server connector

To add a new connector to an AxoRouter host, complete the following steps:

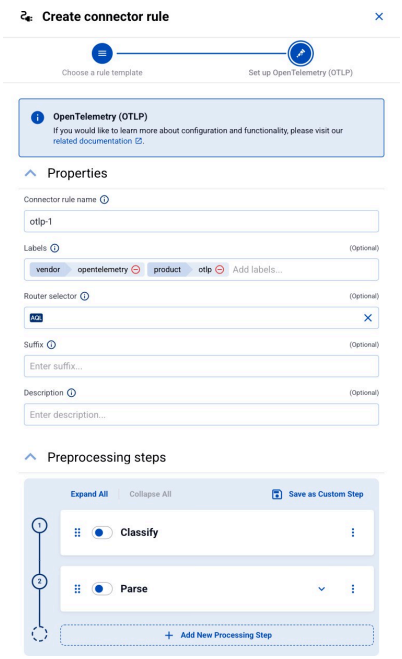
1. Select **Routers > Connector Rules > Add Connector Rule**. (Alternatively, you can select **Add Connector > Add a connector rule** on the **Connectors** page of an AxoRouter host.)



2. Select **Splunk HEC Server**.

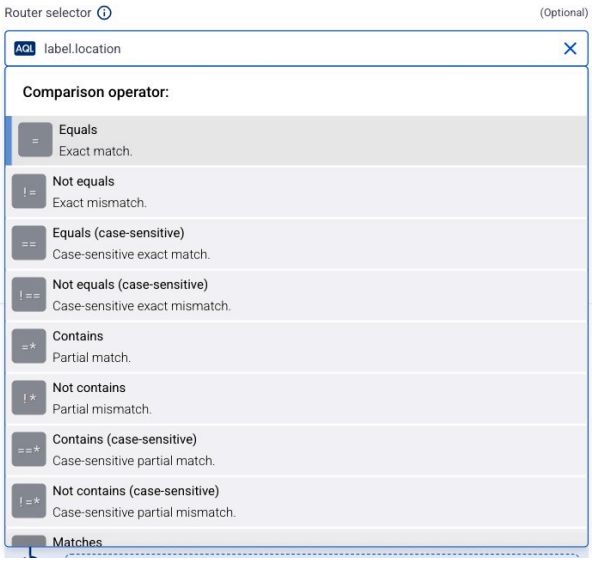
3. Configure the connector rule.

   1. Enter a name for the connector rule into the **Connector rule name** field.



2. (Optional) Add labels to the connector rule. You will be able to use these labels in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps. For details about the message schema and the related connector-specific fields, see the `meta.connector` object.

3. Set the **Router selector** for the connector rule. The selector determines which AxoRouter instances will have a connector based on this connector rule.



You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

- If you leave the router selector empty, it matches every AxoRouter instance.
- To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.

- If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

4. (Optional) Enter a **Suffix** for the connector rule. AxoConsole uses the suffix in the name of the connector instances that it creates on the AxoRouter hosts. The name of the connector instance follows the `<name-of-AxoRouter>-<suffix>` schema. For example, if the name of a matching AxoRouter instance is `axorouter-1`, and the suffix of the rule is `custom`, the name of the connector on that AxoRouter is `axorouter-1-custom`.

   If the **Suffix** field is empty, AxoConsole uses the name of the connector rule instead. For example, if the name of the connector rule is `otlp-1`, the name of the connector is `axorouter-1-otlp-1`.

5. (Optional) Enter a description for the rule.

6. If needed, turn on the **Classify** and **Parse** steps in the **Preprocessing steps** section (they're off by default), so AxoRouter automatically identifies and parses messages sent by supported data sources. If your source is not listed, contact us.

   Enabling these options processes all data received by the connectors created based on this connector rule. If you want to apply classification and parsing more selectively, you can use the **Classify** and **Parse** processing steps in your **Flows**.

   Note that the **Parse** processing step requires **Classify** to be enabled. Parsing automatically parses the data from the content of the message, and replaces the message content (the `log.body` field in the internal message schema) with the structured information.

7. (Optional) If needed, you can add other preprocessing steps to the connector rule. You can use the same processing steps as in flows. These can be useful if you use a dedicated connector rule to fix or annotate data coming from some specific sources. To reuse the preprocessing steps in other connector rules or flows, select **Save as Custom Step**.

4. Enter the HEC **Token** that the clients must present. This option is required, the connector has no unauthenticated mode.

   The clients send this token in the `Authorization: Splunk <token>` header of every request, and AxoRouter refuses the requests that don't carry it.

5. (Optional) Set the **Port** number where the connector receives the requests. If you leave it empty, the connector listens on port `8088` by default.

   CAUTION:

> TCP and UDP port range 29500–29519 is reserved for AxoRouter's internal listeners and cannot be used in connector rules.

6. (Optional) To receive data over HTTPS, select **Enable TLS**, then set the certificate and the private key that AxoRouter shows to the clients.

> **Note**
>
> **These settings apply to every AxoRouter that matches the Router selector of the connector rule. If you paste values for the server certificate and server private key, every matching AxoRouter uses the same shared certificate, which is not recommended. Instead, use a separate certificate for each AxoRouter by setting the file path option, or create a separate connector rule for each AxoRouter.**

   - **Server certificate**: The certificate that AxoRouter shows to the clients.
     - To provide the server certificate from the AxoConsole, select **Paste value** and paste the server certificate. AxoConsole will automatically deploy the server certificate to the related AxoRouters.
     - To provide the server certificate from a file, select **File path** and enter the path where the certificate file is located (for example, `/etc/axorouter/user-config/server-certificate.pem`). The path must be mounted into the container, `/etc/axorouter` is mounted by default. You must manually copy that file to its place on every affected AxoRouter, or you can use the configuration management system of your choice.
   - **Server private key**: The private key of the server certificate.
     - To provide the server private key from the AxoConsole, select **Paste value** and paste the server private key. AxoConsole will automatically deploy the server private key to the related AxoRouters.
     - To provide the server private key from a file, select **File path** and enter the path where the private key file is located (for example, `/etc/axorouter/user-config/server-private-key.pem`). The path must be mounted into the container, `/etc/axorouter` is mounted by default. You must manually copy that file to its place on every affected AxoRouter, or you can use the configuration management system of your choice.
7. (Optional) Select **More options** to change the advanced options of the connector.
8. Select **Add**.

   Axoflow automatically creates connectors on the AxoRouter hosts that match the **Router selector**.

Make sure to enable the ports you've configured in the connector on the firewall of the AxoRouter host, and on other firewalls between the AxoRouter host and your data sources.

9. Send a test request to the connector.

    1. Open the **Overview** tab of the AxoRouter host where you've created the connector and check the IP address or FQDN of the host.

    2. Open a terminal on your computer and send an event to the `/services/collector` endpoint of the connector.

    ```
    curl -X POST -H 'Authorization: Splunk <your-token>' --data
    '{"host":"myhost","event":"sample HEC message"}' http://<router-IP-
    address>:<connector-port-number>/services/collector
    ```

10. Configure your clients to send their data to AxoRouter. Set the HEC endpoint of the client to the address and the port of the connector, and the HEC token to the **Token** of the connector.

> **Note**
>
> **You can also use Log tapping on the AxoRouter host to see the incoming messages.**

Advanced options

Select **More options** on the connector rule to set the following options.

- **Maximum connections**: The maximum number of connections that the connector accepts at the same time.
- **Maximum request size**: The maximum accepted size of a request body, in bytes. AxoRouter refuses the requests that are larger than this.

You can also adjust the flow-control related options.

- **Log window size**: The size (number of messages) of the initial window used in flow control. For details, see the AxoSyslog documentation.

  If you change the value of **Log window size**, the change will affect only new connections, the **Log window size** of already established connections will not change. To apply the new **Log window size** value to every connection, restart the

`axorouter.service` service on the **Services** page of the matching AxoRouters. A simple configuration reload is *NOT* sufficient.

If the connector rule is receiving data using the UDP protocol, always restart the `axorouter.service` service on the **Services** page of the matching AxoRouters.

- **Log fetch limit**: The maximum number of messages fetched from a source during a single poll loop. For details, see the AxoSyslog documentation.

Metadata fields

AxoRouter unwraps the HEC envelope of every request and turns each event into a separate message. It adds the following fields to the `meta` variable:

Analytics label	Message field	Value
`connector.type`	`meta.connector.type`	`splunkHEC`
`connector.name`	`meta.connector.name`	The **Name** of the connector
`connector.port`	`meta.connector.port`	The port number where the connector receives data
`connector.labels.product`	`meta.connector.labels.product`	Default value: `splunk-hec-server`
`connector.labels.vendor`	`meta.connector.labels.vendor`	Default value: `splunk`

AxoRouter reads the following fields of the HEC envelope as well:

- The `event` field becomes `log.body`. The rest of the envelope doesn't appear in the body of the message.
- The `host` field becomes `meta.host.name`.
- The `time` field becomes the message timestamp (epoch seconds, fractional part allowed) `log.time_unix_nano`.
- Other fields are dropped.

You can add further parsing to the body using processing steps in Flows, for example, using FilterX or Regex processing steps.

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

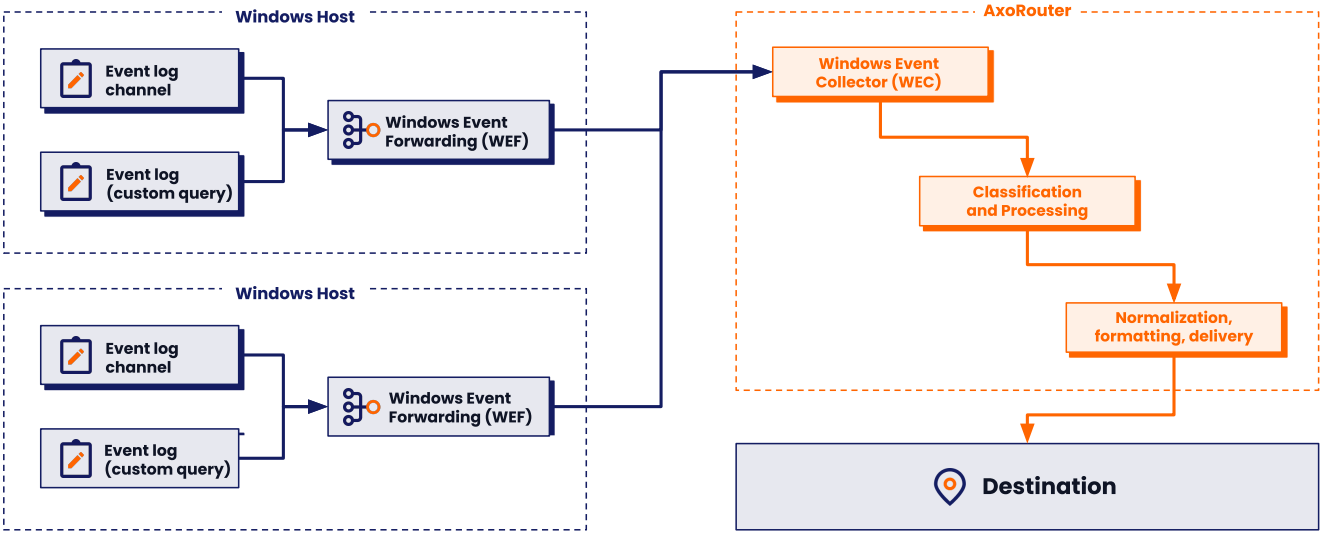
7.11 Windows Event Collector (WEC)

Configure the AxoRouter Windows Events connector to collect logs from Microsoft Windows hosts using Windows Event Forwarding (WEF).

The AxoRouter Windows Events connector can receive Windows Event Logs by running a Windows Event Collector (WEC) server. After enabling the Windows Events connector, you can configure your Microsoft Windows hosts to forward their event logs to AxoRouter using Windows Event Forwarding (WEF).

Windows Event Forwarding (WEF) reads any operational or administrative event logged on a Windows host and forwards the events you choose to a Windows Event Collector (WEC) server, in this case AxoRouter. You can:

- read all data from specific Event Log channels,
- use a custom event filter query, or
- read data from a specific EVTX file.



Prerequisites

When using TLS authentication, you need:

On the AxoRouter host:

- A CA certificate (in PEM format) that AxoRouter uses to authenticate the clients.
- A certificate and the matching private key (in PEM format) that AxoRouter shows to the clients.
- These files must be available on the AxoRouter host and readable by the `axorouter` service for the connector to work.

On the Windows hosts:

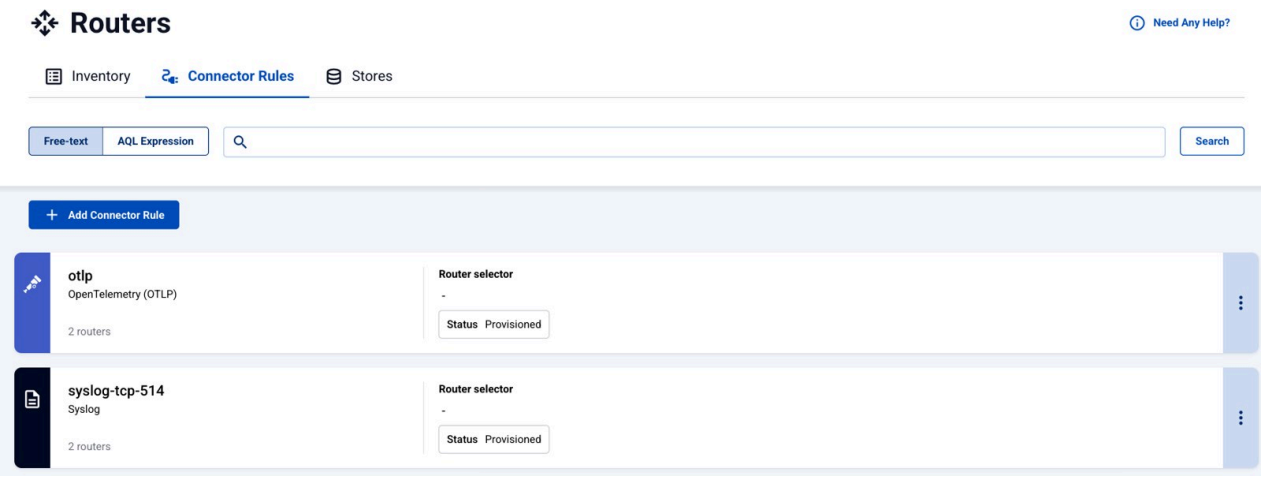
- The certificate of the Windows host must have read permission to the **Network Service**.

Add new Windows Event Log connector

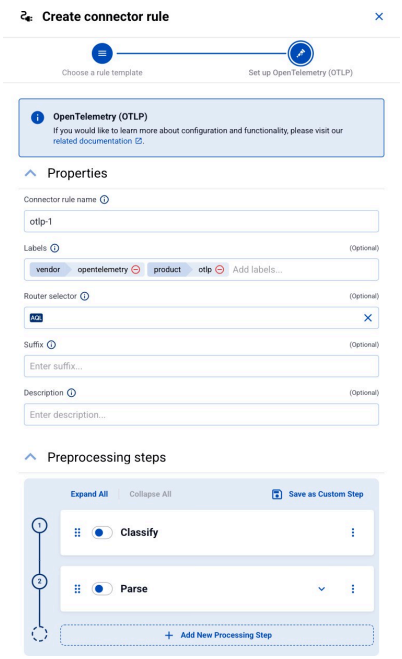
To add a new connector to an AxoRouter host, complete the following steps.

> **Note**
>
> **Currently you can configure one Windows Events connector for an AxoRouter.**

1. Select **Routers > Connector Rules > Add Connector Rule**. (Alternatively, you can select **Add Connector > Add a connector rule** on the **Connectors** page of an AxoRouter host.)

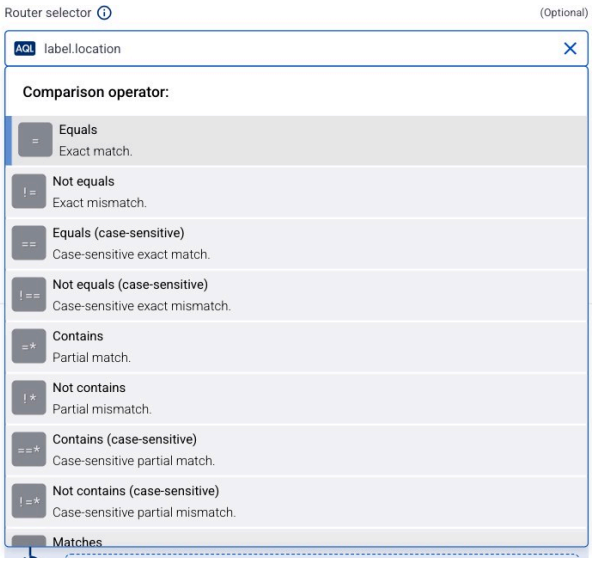


2. Select **Windows Events**.
3. Configure the connector rule.
    1. Enter a name for the connector rule into the **Connector rule name** field.



2. (Optional) Add labels to the connector rule. You will be able to use these labels in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps. For details about the message schema and the related connector-specific fields, see the `meta.connector` object.

3. Set the **Router selector** for the connector rule. The selector determines which AxoRouter instances will have a connector based on this connector rule.



You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

- If you leave the router selector empty, it matches every AxoRouter instance.
- To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.

- If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

4. (Optional) Enter a **Suffix** for the connector rule. AxoConsole uses the suffix in the name of the connector instances that it creates on the AxoRouter hosts. The name of the connector instance follows the `<name-of-AxoRouter>-<suffix>` schema. For example, if the name of a matching AxoRouter instance is `axorouter-1`, and the suffix of the rule is `custom`, the name of the connector on that AxoRouter is `axorouter-1-custom`.

   If the **Suffix** field is empty, AxoConsole uses the name of the connector rule instead. For example, if the name of the connector rule is `otlp-1`, the name of the connector is `axorouter-1-otlp-1`.

5. (Optional) Enter a description for the rule.

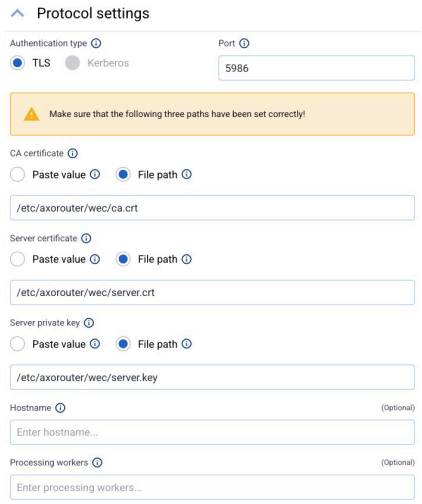
6. If needed, turn on the **Classify** and **Parse** steps in the **Preprocessing steps** section (they're off by default), so AxoRouter automatically identifies and parses messages sent by supported data sources. If your source is not listed, contact us.

   Enabling these options processes all data received by the connectors created based on this connector rule. If you want to apply classification and parsing more selectively, you can use the **Classify** and **Parse** processing steps in your **Flows**.

   Note that the **Parse** processing step requires **Classify** to be enabled. Parsing automatically parses the data from the content of the message, and replaces the message content (the `log.body` field in the internal message schema) with the structured information.

7. (Optional) If needed, you can add other preprocessing steps to the connector rule. You can use the same processing steps as in flows. These can be useful if you use a dedicated connector rule to fix or annotate data coming from some specific sources. To reuse the preprocessing steps in other connector rules or flows, select **Save as Custom Step**.

4. Configure the protocol-level settings of the connector.



1. (Optional) Set the **Hostname** field. Clients address this hostname. Note that:
   - This is the hostname that is configured as the Subscription Manager address in the Group Policy Editor.
   - If unset, the server automatically detects it from the request's host header.
2. (Optional) If for some reason you don't want to run the connector on the default port (`5986`), adjust the **Port** field.

   > **CAUTION:**
   >
   > TCP and UDP port range 29500–29519 is reserved for AxoRouter's internal listeners and cannot be used in connector rules.

3. Set the paths for the certificates and keys used for the TLS-encrypted communication with the clients.

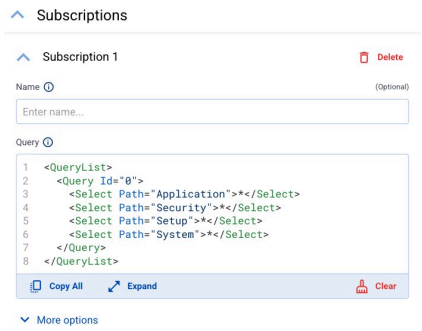
   > ## Note
   >
   > **These settings apply to every AxoRouter that matches the Router selector of the connector rule. If you paste values for the server certificate and server private key, every matching AxoRouter uses the same shared certificate, which is not recommended. Instead, use a separate certificate for each AxoRouter by setting the file path option, or create a separate connector rule for each AxoRouter.**

   - **CA certificate**: The CA certificate that AxoRouter uses to authenticate the clients. The file can contain multiple issuer certificates to allow clients connecting from multiple different domains simultaneously. If you want to limit which clients are accepted, set the **More options > Certificate subject filter** field.

- To provide the CA certificate from the AxoConsole, select **Paste value** and paste the CA certificate. AxoConsole will automatically deploy the CA certificate to the related AxoRouters.
- To provide the CA certificate from a file, select **File path** and enter the path where the certificate file is located (for example, `/etc/axorouter/user-config/ca-certificate.pem`). The path must be mounted into the container, `/etc/axorouter` is mounted by default. You must manually copy that file to its place on every affected AxoRouter, or you can use the configuration management system of your choice.

- **Server certificate**: The certificate that AxoRouter shows to the clients. The Common Name must match the hostname used by the clients to contact the server.
  - To provide the server certificate from the AxoConsole, select **Paste value** and paste the server certificate. AxoConsole will automatically deploy the server certificate to the related AxoRouters.
  - To provide the server certificate from a file, select **File path** and enter the path where the certificate file is located (for example, `/etc/axorouter/user-config/server-certificate.pem`). The path must be mounted into the container, `/etc/axorouter` is mounted by default. You must manually copy that file to its place on every affected AxoRouter, or you can use the configuration management system of your choice.
- **Server private key**: The private key of the server certificate.
  - To provide the server private key from the AxoConsole, select **Paste value** and paste the server private key. AxoConsole will automatically deploy the server private key to the related AxoRouters.
  - To provide the server private key from a file, select **File path** and enter the path where the private key file is located (for example, `/etc/axorouter/user-config/server-private-key.pem`). The path must be mounted into the container, `/etc/axorouter` is mounted by default. You must manually copy that file to its place on every affected AxoRouter, or you can use the configuration management system of your choice.

4. (Optional) If needed, set the number of workers.
   - **Processing workers**: Number of worker threads to use to process the messages received by the source connector. Note that this enables parallel processing, which might cause that the messages will be processed out of order. Increasing the number of worker threads can drastically improve the performance of the source. For details, see the AxoSyslog documentation.
5. Configure the subscriptions of the connector.



1. Select **Add new Subscription**.
2. (Optional) Set a name for the subscription. If you leave it empty, AxoConsole automatically generates a name.
3. Enter the event filter query into the **Query** field. This query specifies which events are collected by the subscription. For details on the query syntax, see the Microsoft documentation.

   A single query can retrieve events from a maximum of 256 different channels.

   For example, the following example queries every event from the Security, System, Application, and Setup channels.

   ```
   <QueryList>
     <Query Id="0">
       <Select Path="Application">*</Select>
       <Select Path="Security">*</Select>
       <Select Path="Setup">*</Select>
       <Select Path="System">*</Select>
     </Query>
   </QueryList>
   ```

4. (Optional) If needed, you can configure other low-level options in the **More options** section. For details, see Additional options.

6. Select **Add**.

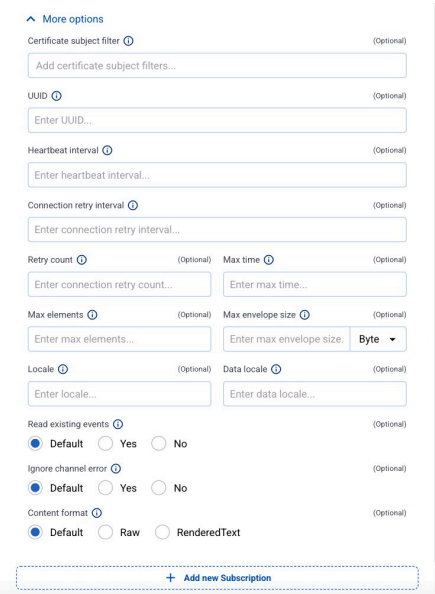
   Axoflow automatically creates connectors on the AxoRouter hosts that match the **Router selector**.

   Make sure to enable the ports you've configured in the connector on the firewall of the AxoRouter host, and on other firewalls between the AxoRouter host and your data sources.

7. Configure Windows Event Forwarding on your Windows clients.

Additional options

You can set the following options of the WEC connector under **Subscriptions > More options**.



- **Certificate subject filter**: A simple string to filter the clients based on the Common Name of their certificate. You can use the `*` and `?` wildcard characters.
- **UUID**: A unique ID for the subscription. If empty, AxoConsole automatically generates it.
- **Heartbeat interval**: The number of seconds before the client sends a heartbeat message. The client sends heartbeat messages if it has no new events to send. Default value: `3600s`
- **Connection retry interval**: Time between reconnection attempts. Default value: `60s`
- **Retry count**: Number of times the client attempts to reconnect if AxoRouter is unreachable. Default value: `10`
- **Max time**: The maximum number of seconds the client aggregates new events before sending them in a batch. Default value: `30s`
- **Max elements**: The maximum number of events that the client aggregates before sending them in a batch. By default, it's empty, meaning that only the **Max time** and **Max envelope size** options limit the aggregation. Default value: empty
- **Max envelope size**: The maximum number of bytes in the SOAP envelope used to deliver the events. Default value: `512000 bytes`
- **Locale**: The language in which rendering information is expected, for example, `en-US`. Default value: `Client choose`

- **Data locale**: The language in which numerical data is expected to be formatted, for example, `en-US`. Default value: `Client choose`
- **Read existing events**: If enabled (**Yes**), the event source sends:
  - all existing events that match the filter, and
  - any events that subsequently occur for that event source.

  If disabled (**No**), existing events are ignored.

  Default value: **No**
- **Ignore channel error**: Subscription queries that result in errors terminate the processing of the clients. Enable this option to ignore such errors. Default value: **Yes**
- **Content format**: Determines whether to include rendering information (`RenderedText`) with events or not (`Raw`). Default value: **Raw**

Configure Windows Event Forwarding

To configure Windows Event Forwarding (WEF) on your Windows clients to forward their events to the AxoRouter WEC connector, complete the following steps.

1. Configure the host so it can validate the certificate of AxoRouter.
   1. Open the Microsoft Management Console (`mmc.exe`).
   2. Select **File > Add/Remove Snap-ins**.
   3. Add the **Certificates** snap-in.
   4. Select **Computer Account**.
   5. Right-click **Personal**, then select **All Tasks > Import**.
   6. Select and import the client certificate file (`.p12`). The Windows client shows this certificate to AxoRouter.
   7. Select the client certificate, then select **All Tasks > Manage Private Keys**. Give the certificate read permission to the **Network Service**.
   8. Import the CA certificate used to issue the certificate of AxoRouter, then move it to **Trusted Root Certification Authorities**.
   9. Save the Thumbprint of the CA. You'll need it for configuring subscriptions in the Subscription Manager.
2. Set forwarding security data.
   1. Open `CompMgmt.msc` .
      - On domain controller hosts, navigate to **Active Directory Users and Computers**, select your domain, then select **Builtin**.

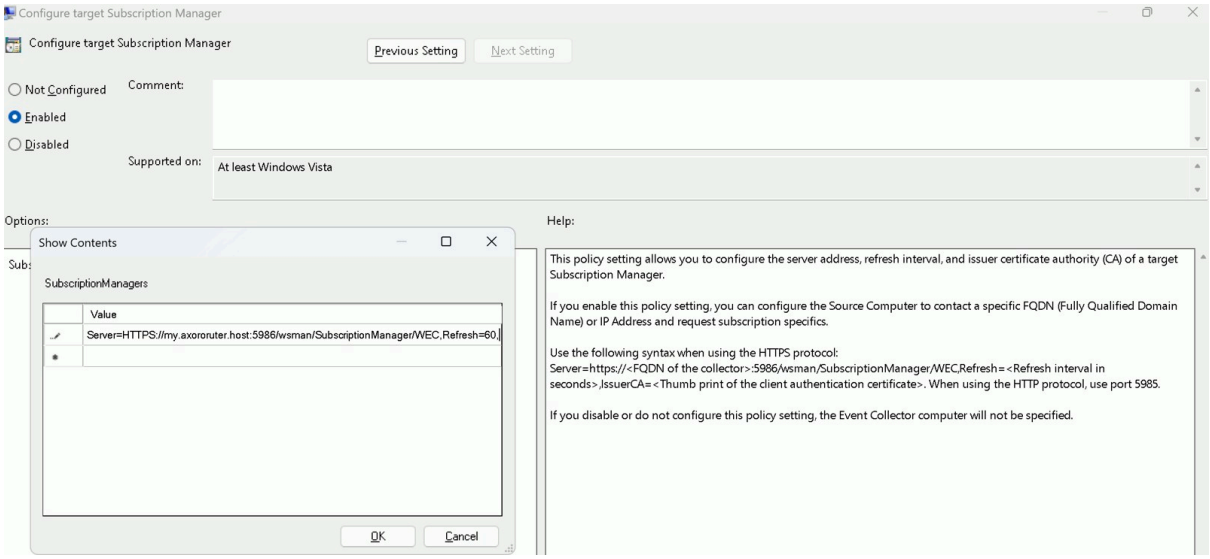
- On generic Windows hosts, navigate to **Local Users and Groups > Groups > Event Log Readers**, then open **Event Log Readers Properties**.

2. Add the `NETWORK SERVICE` account to the **Event Log Readers** group.
3. Reboot the host.
4. Open a command prompt with elevated privileges and run the following commands.

```
winrm qc -q
winrm set winrm/config/client/auth @{Certificate="true"}
```

3. Configure the Subscription Manager.
    1. Open `gpedit.msc`.
    2. Select **Computer Configuration > Administrative Templates > Windows Components node > Event Forwarding**.
    3. Select **SubscriptionManagers**, then select **Enabled**.
    4. Select **Show**.
    5. Add a subscription. Use the hostname or IP address of AxoRouter you've set in the connector. For example: `Server=https://<FQDN-of-AxoRouter>:5986/wsman/SubscriptionManager/WEC,Refresh=<Refresh-interval-in-seconds>,IssuerCA=<thumbprint-of-AxoRouters-root-CA>`

       The refresh interval determines how often the client checks for changes in the subscriptions. We recommend using 30 seconds during a test period (while you're fine-tuning your subscriptions), and an hour (`3600`) or longer in production environments.



During testing and troubleshooting, the **Applications and Services Logs\Microsoft\Windows\Eventlog-ForwardingPlugin** and **Applications and Services Logs\Microsoft\Windows\Windows Remote Management** event log channels can be useful.

Metadata fields

The AxoRouter Windows Events connector adds the following fields to the `meta` variable:

Analytics label	Message field	Value
`connector.type`	`meta.connector.type`	`windowsEvents`
`connector.name`	`meta.connector.name`	The **Name** of the connector
`connector.port`	`meta.connector.port`	The port number where the connector receives data

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

7.12 Vendors

Onboard vendor-specific source devices and hosts to Axoflow so they appear on the Topology page, send logs to AxoRouter, and report host metrics.

Prerequisites

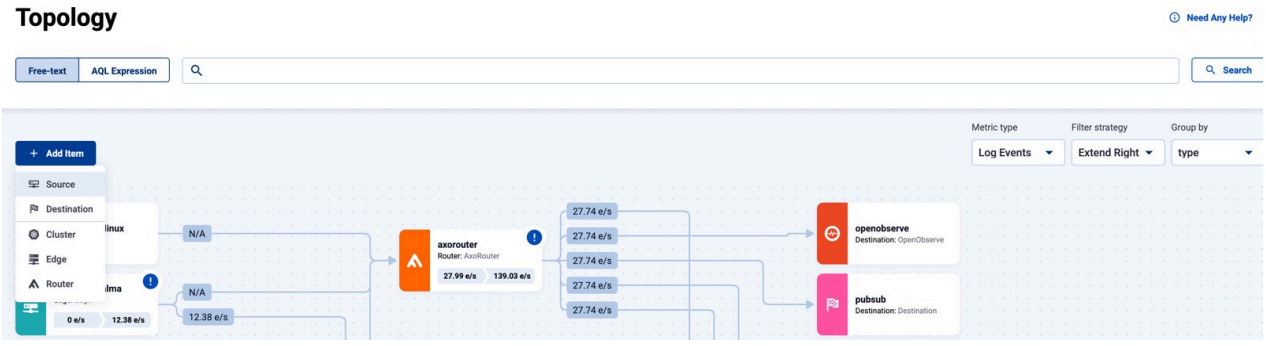
- You have administrative access to the source device or host.
- You have an AxoRouter deployed and configured with a Syslog connector that has parsing and classification enabled (by default, every AxoRouter has such connectors). This device is going to receive the data from the source device or host.
- You know the IP address the AxoRouter. To find it:

  1. Open the AxoConsole.
  2. Select the **Routers** or the **Topology** page.
  3. Select on AxoRouter instance that is going to receive the logs.
  4. Check the **Networks > Address** field.

Onboard a vendor source

To onboard sources that are specifically supported by Axoflow, complete the following steps. Onboarding allows you to collect metrics about the host, and display the host on the **Topology** page.

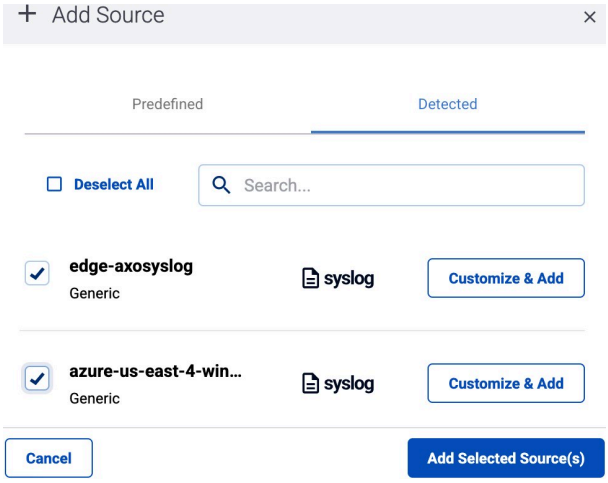
1. Open the AxoConsole.
2. Select **Topology**.
3. Select **Add Item > Source**.



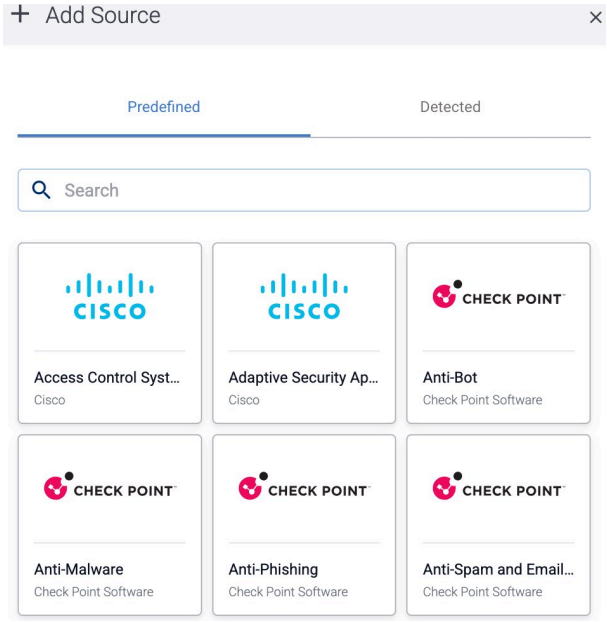
4. Add the source to AxoConsole using one of the following methods:

   - If the source is already sending logs to an AxoRouter instance that is registered in the AxoConsole, select **Detected**, then select the source.

     You can add multiple detected sources in a single step in bulk.

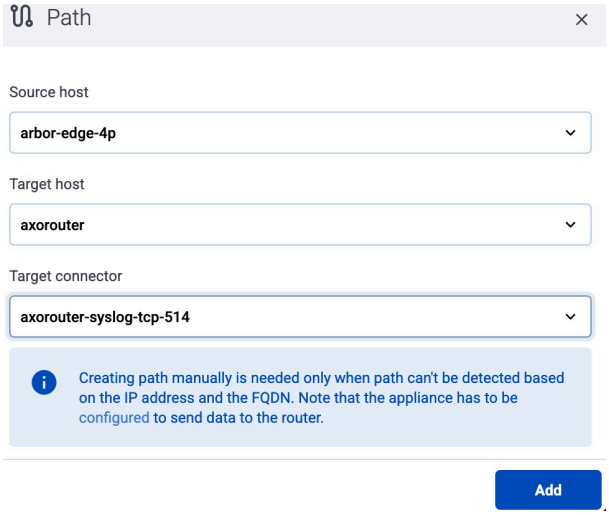


- Otherwise, select the type of the source you want to onboard, and follow the on-screen instructions.



5. Connect the source to the destination or AxoRouter instance it's sending logs to. If you've added the source from the **Detected** list, you can skip this step, as the path is created automatically.

    1. Select **Topology > Add Item > Path**.



2. If not set automatically, select the beginning of the path in the **Source host** field.
3. Select the target router or aggregator this source is sending its data to in the **Target host** field, for example, `axorouter`.
4. Select the **Target connector**. The connector determines how the destination receives the data (for example, using which protocol or port).
5. Select **Add**. The new path appears on the **Topology** page.



6. If you haven't already done so, configure the source to send logs to an AxoRouter instance. For vendor-specific configuration steps, default metadata (labels), and SIEM-specific metadata, see the page for each source.

> ## Note
>
> Unless instructed otherwise, configure your source to send the logs to the Syslog connector of AxoRouter, using the appropriate port. Use RFC5424 if the source supports it.
>
> - 514 UDP and TCP for RFC3164 (BSD-syslog) and RFC5424 (IETF-syslog) formatted traffic. AxoRouter automatically recognizes and handles both formats.
> - 601 TCP (RFC 3195 - Reliable Delivery for syslog) for RFC5424 (IETF-syslog) and RFC3164 (BSD-syslog) formatted traffic. AxoRouter automatically recognizes and handles both formats.
> - 6514 TCP for TLS-encrypted syslog traffic (RFC 5425).

7. If the source doesn't show up on the **Topology** page, see A source doesn't appear on the Topology page.

7.12.1 A10 Networks

Send logs from A10 Networks vThunder application delivery controllers to Axoflow.

7.12.1.1 vThunder

Send A10 Networks vThunder logs to Axoflow, and look up the labels that Axoflow adds to its data.

vThunder: Delivers application load balancing, traffic management, and DDoS protection for enterprise networks.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`a10networks`
`product`	`meta.product`	`vthunder`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	source	index
`a10networks:vThunder:cef`	`a10networks:vThunder`	`netwaf`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `A10_LOAD_BALANCER`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.2 Amazon

Collect logs from Amazon CloudWatch and forward them to Axoflow for processing.

7.12.2.1 AWS CloudTrail

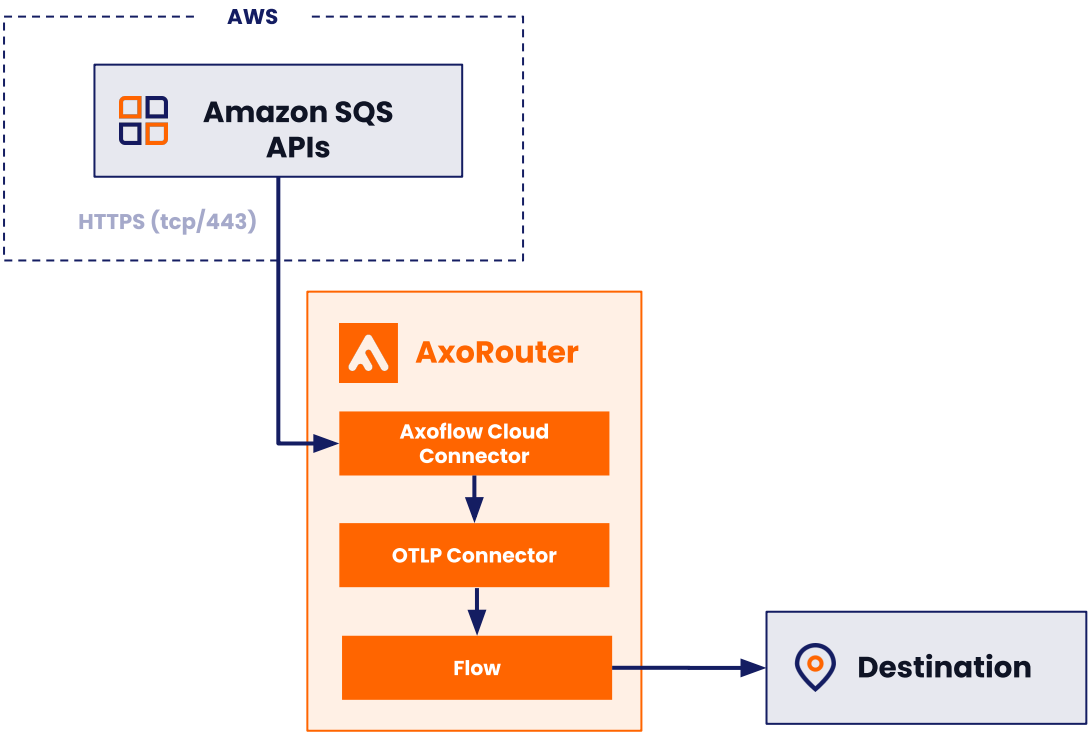
Collect the CloudTrail log files that your trail delivers to an Amazon S3 bucket with a Cloud Connector, and look up the labels Axoflow adds.

AWS CloudTrail: Tracks AWS account activity and API usage for auditing, governance, and compliance monitoring.

Axoflow can collect the log files that your AWS CloudTrail trail delivers to an Amazon S3 bucket. At a high level, the process looks like this:

- Create a **Connector Rule** on the AxoRouter that deploys an Axoflow Cloud Connector and collects the data from the S3 bucket that receives the trail log files. Axoflow Cloud Connector is a simple container that runs on an AxoRouter instance.
- Configure a Flow on AxoConsole that processes and routes the collected data to your destination (for example, Splunk or another SIEM).

The Axoflow Cloud Connector doesn't poll the bucket. Instead, it reads the Amazon SQS queue that receives the `s3:ObjectCreated:*` event notifications of the bucket, and downloads every object that the queue announces.



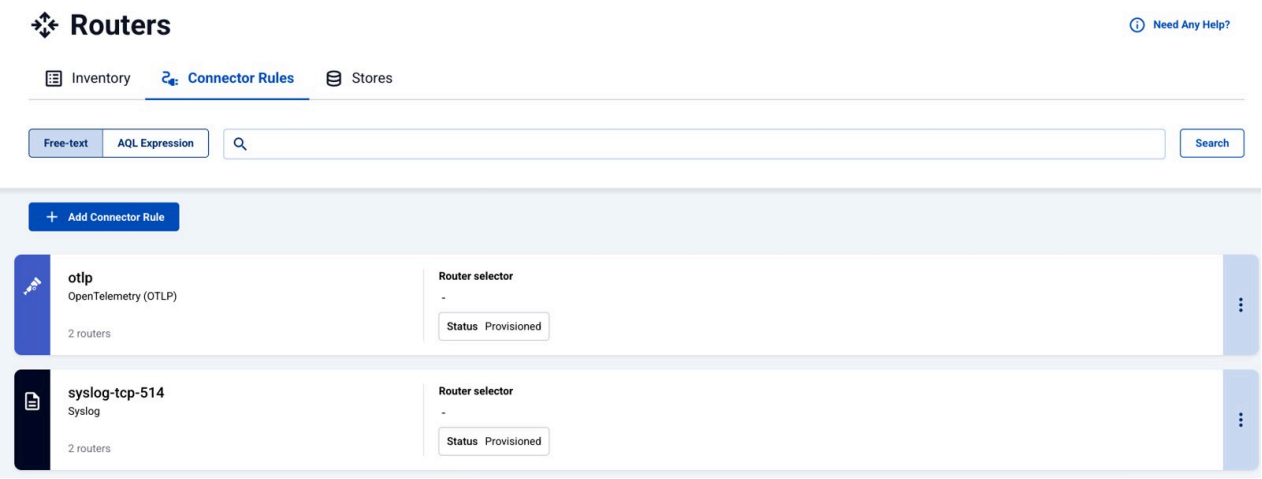
Prerequisites

- An AxoRouter instance that:
  - Was provisioned with **Enable Cloudconnectors** turned on.
  - Can reach the Amazon S3 and Amazon SQS endpoints of the region on TCP port 443. Adjust your firewall rules and the access rules of the data source as needed.
  - You know the IP address the AxoRouter. To find it:
    1. Open the AxoConsole.
    2. Select the **Routers** or the **Topology** page.
    3. Select on AxoRouter instance that is going to receive the logs.
    4. Check the **Networks > Address** field.
- An AWS account with an active subscription.
- A trail that delivers its log files to an S3 bucket. Axoflow Cloud Connector reads plain and gzip-compressed `.json` objects.
- An SQS queue that receives the `s3:ObjectCreated:*` event notifications of the bucket, either directly or through an SNS topic. The queue must be in the same region as the bucket, because S3 event notifications can only target a queue in the region of the bucket.
- AWS credentials that can read the queue and download the objects of the bucket. Axoflow Cloud Connector can use:
  - a static access key and secret key,
  - the default credential chain of the host it runs on (for example, an EC2 instance role, EKS IRSA or Pod Identity, or environment variables),
  - an IAM role that it assumes with either of the above.

Configure Axoflow Cloud Connector on AxoRouter

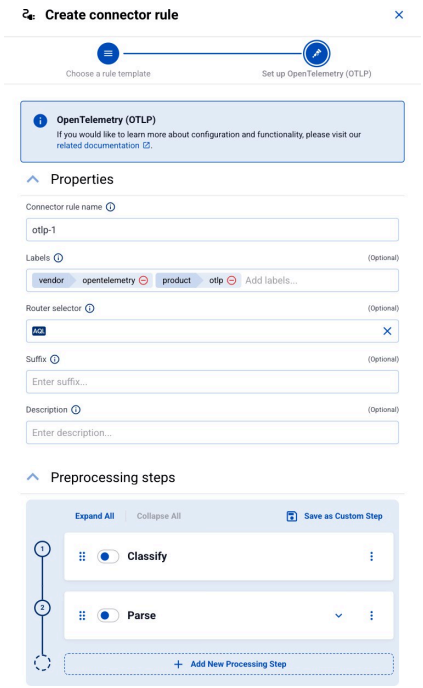
To collect data from CloudTrail, complete the following steps.

1. Select **Routers > Connector Rules > Add Connector Rule**. (Alternatively, you can select **Add Connector > Add a connector rule** on the **Connectors** page of an AxoRouter host.)



2. Select the **AWS CloudTrail**.

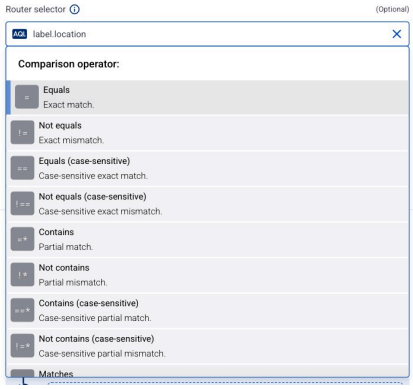
3. Enter a name for the connector rule into the **Connector rule name** field.



4. (Optional) Add labels to the connector rule. You will be able to use these labels in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps. For details about the message schema and the related connector-specific fields, see the `meta.connector` object.

5. Set the **Router selector** for the connector rule. The selector determines which AxoRouter instances get a connector from this connector rule.

> **CAUTION:**
>
> Each connector rule for AWS CloudTrail must match exactly one AxoRouter. Use the `name` or `guid` AQL selector to make sure it matches only one AxoRouter. Otherwise several Axoflow Cloud Connector instances collect the same data, and Axoflow receives every message multiple times.



You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

- If you leave the router selector empty, it matches every AxoRouter instance.
- To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.
- If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

6. (Optional) Enter a **Suffix** for the connector rule. AxoConsole uses the suffix in the name of the connector instances that it creates on the AxoRouter hosts. The name of the connector instance follows the `<name-of-AxoRouter>-<suffix>` schema. For example, if the name of a matching AxoRouter instance is `axorouter-1`, and the suffix of the rule is `custom`, the name of the connector on that AxoRouter is `axorouter-1-custom`.

   If the **Suffix** field is empty, AxoConsole uses the name of the connector rule instead. For example, if the name of the connector rule is `otlp-1`, the name of the connector is `axorouter-1-otlp-1`.

7. (Optional) Enter a description for the rule.

8. Configure how Axoflow Cloud Connector reaches the bucket and the queue.

   - **Region**: The AWS region of the S3 bucket and the SQS queue. S3 event notifications can only target a queue in the region of the bucket.
   - **Bucket**: The name of the S3 bucket that CloudTrail delivers its log files to.
   - **SQS queue URL**: The HTTPS URL of the SQS queue that receives the `s3:ObjectCreated:*` event notifications of the bucket.

   CAUTION:

   Use the event notification of the bucket, not the SNS topic that CloudTrail itself can publish to. The CloudTrail notification has a different payload, and Axoflow Cloud Connector doesn't process it.

9. (Optional) Select **More options** to change the advanced options of the connector, for example, to set the credentials that Axoflow Cloud Connector authenticates with.
10. Select **Add**. The Axoflow Cloud Connector starts reading the queue and forwarding the collected log files to the AxoRouter instance.
11. Verify that the Axoflow Cloud Connector is running.
    1. Select **Routers**, then select the AxoRouter you added the Axoflow Cloud Connector to.
    2. Select **Services** and check that **axoflow-cloudconnector.service** is running without errors.
12. Create a Flow to route the data from the AxoRouter instance to a destination. You can use the Labels of this source to select its messages.

Advanced options

Select **More options** on the connector rule to set the following options.

- **Prefix**: Collect only the objects whose key starts with this prefix. It's useful if the bucket holds other log types as well.
- **Access key** and **Secret key**: A static AWS credential. Set both options, or neither of them. If you leave them empty, Axoflow Cloud Connector uses the default AWS credential chain of the host it runs on, for example, an EC2 instance role, EKS IRSA or Pod Identity, or environment variables.
- **Role ARN**: The ARN of an IAM role that Axoflow Cloud Connector assumes before it reads the queue and the bucket, for example, when the log files are delivered to the bucket of another account. Axoflow Cloud Connector assumes the role with the static keys if you set them, otherwise with the identity of the host.
- **External ID**: The external ID (`sts:ExternalId`) that the trust policy of the role requires. Set it only together with **Role ARN**.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`amazon`
`product`	`meta.product`	`aws-cloudtrail`

`service`	`meta.service.name`	`management` , `data` , `insight` , or `networkactivity`

The `service` label depends on the category of the CloudTrail event (the `eventCategory` field): `Management` , `Data` , `Insight` , or `NetworkActivity` .

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`aws:cloudtrail`	`aws-activity`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `AWS_CLOUDTRAIL` .

7.12.2.2 AWS VPC Flow Logs

Collect VPC flow logs from an Amazon S3 bucket (for example, an AWS Security Lake subscription) with a Cloud Connector.

AWS VPC Flow Logs: Records the IP traffic going to and from network interfaces in a VPC.

Axoflow can collect the AWS VPC Flow Logs that your VPC flow log subscription (for example, an AWS Security Lake subscription) delivers to an Amazon S3 bucket. At a high level, the process looks like this:

- Create a **Connector Rule** on the AxoRouter that deploys an Axoflow Cloud Connector and collects the data from the S3 bucket that receives the flow log files. Axoflow Cloud Connector is a simple container that runs on an AxoRouter instance.
- Configure a Flow on AxoConsole that processes and routes the collected data to your destination (for example, Splunk or another SIEM).

The Axoflow Cloud Connector doesn't poll the bucket. Instead, it reads the Amazon SQS queue that receives the `s3:ObjectCreated:*` event notifications of the bucket, and downloads every object that the queue announces.

Prerequisites

- An AxoRouter instance that:
  - Was provisioned with **Enable Cloudconnectors** turned on.
  - Can reach the Amazon S3 and Amazon SQS endpoints of the region on TCP port 443. Adjust your firewall rules and the access rules of the data source as needed.
  - You know the IP address the AxoRouter. To find it:
    1. Open the AxoConsole.
    2. Select the **Routers** or the **Topology** page.
    3. Select on AxoRouter instance that is going to receive the logs.
    4. Check the **Networks > Address** field.
- An AWS account with an active subscription.
- A VPC flow log subscription that delivers its log files to an S3 bucket in **plain text** format. Axoflow Cloud Connector reads plain and gzip-compressed `.log` objects, but it can't read flow logs delivered in Parquet format.
- An SQS queue that receives the `s3:ObjectCreated:*` event notifications of the bucket, either directly or through an SNS topic. The queue must be in the same region as the bucket, because S3 event notifications can only target a queue in the region of the

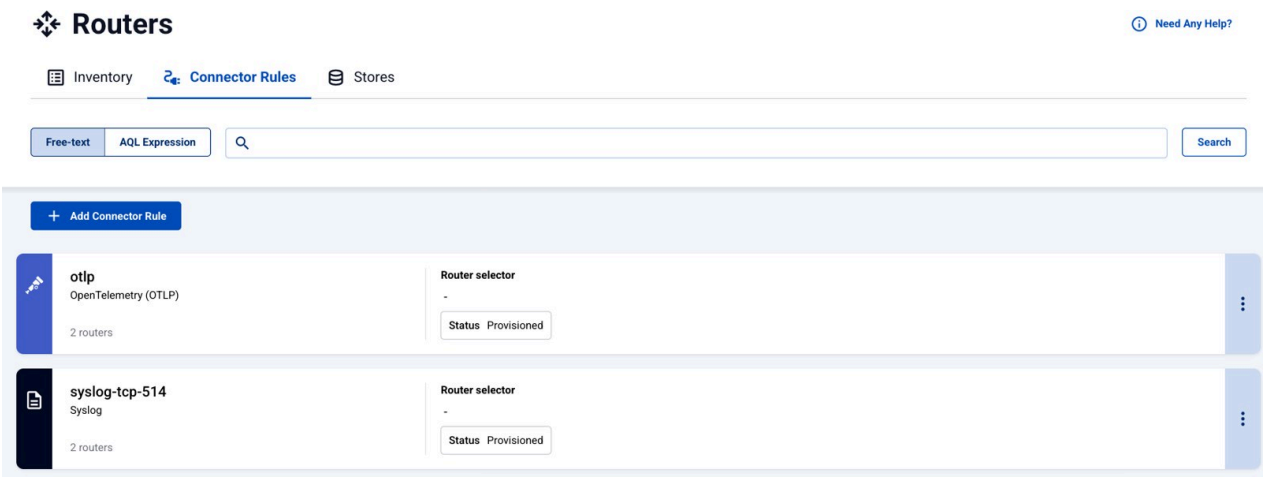
bucket.

- AWS credentials that can read the queue and download the objects of the bucket. Axoflow Cloud Connector can use:
  - a static access key and secret key,
  - the default credential chain of the host it runs on (for example, an EC2 instance role, EKS IRSA or Pod Identity, or environment variables),
  - an IAM role that it assumes with either of the above.

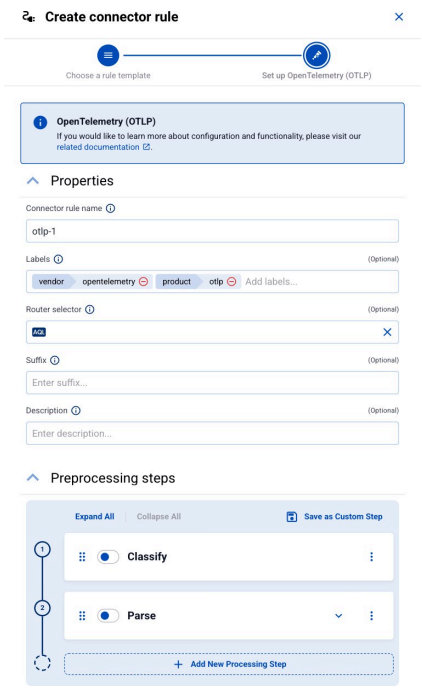
Configure Axoflow Cloud Connector on AxoRouter

To collect VPC flow logs, complete the following steps.

1. Select **Routers > Connector Rules > Add Connector Rule**. (Alternatively, you can select **Add Connector > Add a connector rule** on the **Connectors** page of an AxoRouter host.)



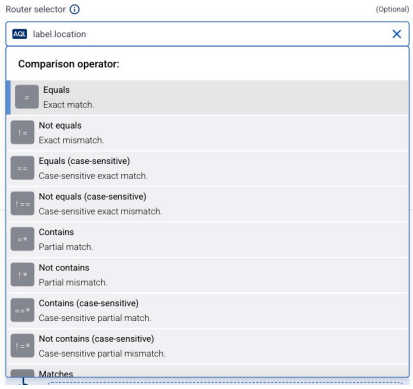
2. Select the **AWS VPC Flow Logs**.
3. Enter a name for the connector rule into the **Connector rule name** field.



4. (Optional) Add labels to the connector rule. You will be able to use these labels in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps. For details about the message schema and the related connector-specific fields, see the `meta.connector` object.

5. Set the **Router selector** for the connector rule. The selector determines which AxoRouter instances get a connector from this connector rule.

   > **CAUTION:**
   >
   > Each connector rule for AWS VPC Flow Logs must match exactly one AxoRouter. Use the `name` or `guid` AQL selector to make sure it matches only one AxoRouter. Otherwise several Axoflow Cloud Connector instances collect the same data, and Axoflow receives every message multiple times.



   You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

   - If you leave the router selector empty, it matches every AxoRouter instance.

- To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.
- If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

6. (Optional) Enter a **Suffix** for the connector rule. AxoConsole uses the suffix in the name of the connector instances that it creates on the AxoRouter hosts. The name of the connector instance follows the `<name-of-AxoRouter>-<suffix>` schema. For example, if the name of a matching AxoRouter instance is `axorouter-1`, and the suffix of the rule is `custom`, the name of the connector on that AxoRouter is `axorouter-1-custom`.

   If the **Suffix** field is empty, AxoConsole uses the name of the connector rule instead. For example, if the name of the connector rule is `otlp-1`, the name of the connector is `axorouter-1-otlp-1`.

7. (Optional) Enter a description for the rule.
8. Configure how Axoflow Cloud Connector reaches the bucket and the queue.

   - **Region**: The AWS region of the S3 bucket and the SQS queue. S3 event notifications can only target a queue in the region of the bucket.
   - **Bucket**: The name of the S3 bucket that the VPC flow log subscription delivers its log files to.
   - **SQS queue URL**: The HTTPS URL of the SQS queue that receives the `s3:ObjectCreated:*` event notifications of the bucket.

9. (Optional) Select **More options** to change the advanced options of the connector, for example, to set the credentials that Axoflow Cloud Connector authenticates with.
10. Select **Add**. The Axoflow Cloud Connector starts reading the queue and forwarding the collected flow logs to the AxoRouter instance.
11. Verify that the Axoflow Cloud Connector is running.

    1. Select **Routers**, then select the AxoRouter you added the Axoflow Cloud Connector to.
    2. Select **Services** and check that **axoflow-cloudconnector.service** is running without errors.

12. Create a Flow to route the data from the AxoRouter instance to a destination. You can use the Labels of this source to select its messages.

Advanced options

Select **More options** on the connector rule to set the following options.

- **Prefix**: Collect only the objects whose key starts with this prefix. It's useful if the bucket holds other log types as well.
- **Access key** and **Secret key**: A static AWS credential. Set both options, or neither of them. If you leave them empty, Axoflow Cloud Connector uses the default AWS credential chain of the host it runs on, for example, an EC2 instance role, EKS IRSA or Pod Identity, or environment variables.
- **Role ARN**: The ARN of an IAM role that Axoflow Cloud Connector assumes before it reads the queue and the bucket, for example, when the log files are delivered to the bucket of another account. Axoflow Cloud Connector assumes the role with the static keys if you set them, otherwise with the identity of the host.
- **External ID**: The external ID (`sts:ExternalId`) that the trust policy of the role requires. Set it only together with **Role ARN**.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`amazon`
`product`	`meta.product`	`aws-vpc-flow-logs`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`aws:cloudwatchlogs:vpcflow`	`aws-activity`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `AWS_VPC_FLOW` .

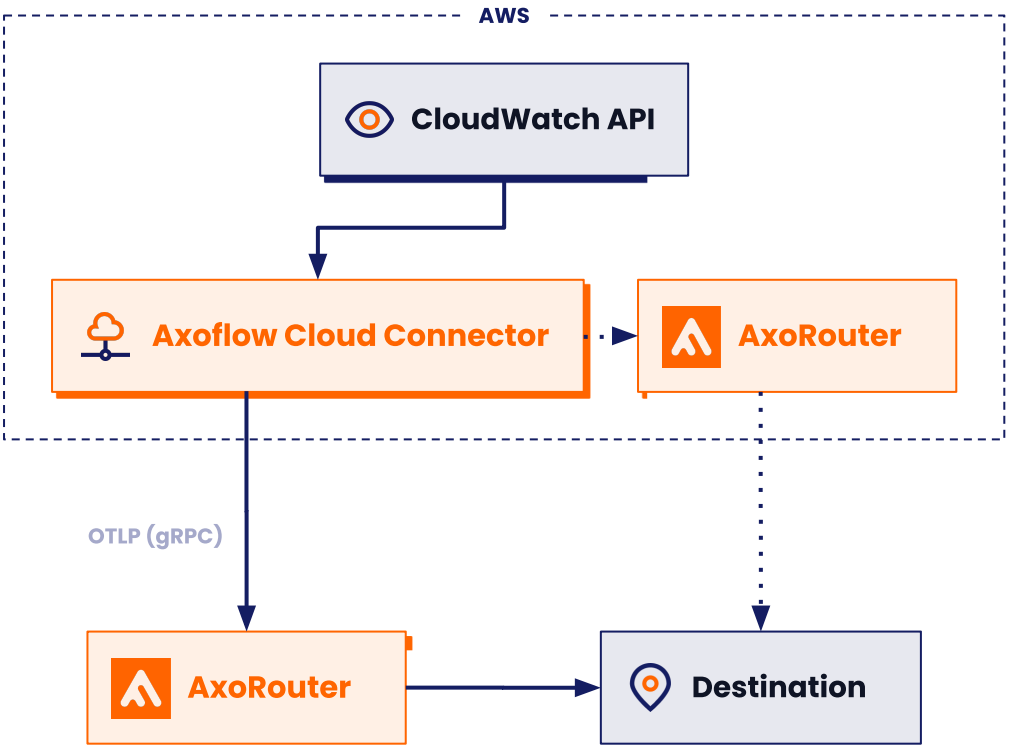
7.12.2.3 CloudWatch

Collect data from Amazon CloudWatch, and look up the labels that Axoflow adds to its data.

CloudWatch: Monitors AWS resources and applications by collecting metrics, logs, and setting alarms.

Axoflow can collect data from your Amazon CloudWatch. At a high level, the process looks like this:

- Deploy an Axoflow Cloud Connector that will collect the data from your CloudWatch. Axoflow Cloud Connector is a simple container that you can deploy into AWS, another cloud provider, or on-prem.
- The connector forwards the collected data to the OpenTelemetry connector of an AxoRouter instance. This AxoRouter can be deployed within AWS, another cloud provider, or on-prem.
- Configure a Flow on AxoConsole that processes and routes the collected data to your destination (for example, Splunk or another SIEM).



Prerequisites

- An AWS account with an active subscription.
- A virtual machine or Kubernetes node running to deploy Axoflow Cloud Connector on.

- An AxoRouter instance that can receive data from the connector. Verify that it has an OpenTelemetry Connector (it's enabled by default).
- You know the IP address the AxoRouter. To find it:

  1. Open the AxoConsole.
  2. Select the **Routers** or the **Topology** page.
  3. Select on AxoRouter instance that is going to receive the logs.
  4. Check the **Networks > Address** field.
- The Axoflow Cloud Connector must be able to access the AxoRouter on the port the OpenTelemetry Connector is listening on (by default, port 4317). Depending on where the Axoflow Cloud Connector and AxoRouter are deployed, you might need to adjust firewall and ingress/egress rules in your environment.
- Depending on how you want to authenticate Axoflow Cloud Connector, you'll need an AWS_PROFILE or AWS access keys.

Steps

To collect data from AWS CloudWatch, complete the following steps.

1. Deploy an Axoflow Cloud Connector.

   1. Access the Kubernetes node or virtual machine where you want to deploy Axoflow Cloud Connector.

   2. Set the following environment variable to the IP address of the AxoRouter where you want to forward the data from CloudWatch. This IP address must be accessible from the connector. You can find the IP address of AxoRouter on the **Routers > AxoRouter > Overview** page.

      ```
      export AXOROUTER_ENDPOINT=<AxoRouter-IP-address>
      ```

   3. (Optional) By default, the connector stores positional and other persistence-related data in the `/etc/axoflow-otel-collector/storage` directory. In case you want to use a different directory, set the `STORAGE_DIRECTORY` environment variable.

   4. Run the following command to generate a UUID for the connector. AxoConsole will use this ID to identify the connector.

```
UUID_FULL=$(uuidgen 2>/dev/null || cat /proc/sys/kernel/random/uuid
2>/dev/null || python3 -c "import uuid; print(uuid.uuid4())")
export AXOCLOUDCONNECTOR_DEVICE_ID=$(echo "$UUID_FULL" | cut -d'-' -f1)
```

5. Set TLS encryption to secure the communication between Axoflow Cloud Connector and AxoRouter.

   Configure the TLS-related settings of Axoflow Cloud Connector using the following environment variables.

Variable	Required	Default	Description
`AXOROUTER_TLS_INSECURE`	No	`false`	Disables TLS encryption if set to `true`
`AXOROUTER_TLS_INCLUDE_SYSTEM_CA_CERTS_POOL`	No	`false`	Set to `true` to use the system CA certificates
`AXOROUTER_TLS_CA_FILE`	No	-	Path to the CA certificate file used to validate the certificate of AxoRouter
`AXOROUTER_TLS_CA_PEM`	No	-	PEM-encoded CA certificate
`AXOROUTER_TLS_INSECURE_SKIP_VERIFY`	No	`false`	Set to `true` to disable TLS certificate verification of AxoRouter
`AXOROUTER_TLS_CERT_FILE`	No	-	Path to the certificate file of Axoflow Cloud Connector
`AXOROUTER_TLS_CERT_PEM`	No	-	PEM-encoded client certificate
`AXOROUTER_TLS_KEY_FILE`	No	-	Path to the client private key file of Axoflow Cloud Connector
`AXOROUTER_TLS_KEY_PEM`	No	-	PEM-encoded client private key
`AXOROUTER_TLS_MIN_VERSION`	No	`1.2`	Minimum TLS version to use
`AXOROUTER_TLS_MAX_VERSION`	No	-	Maximum TLS version to use

> **Note**
>
> **You'll have to include the TLS-related environment variables you set in the docker command used to deploy Axoflow Cloud Connector.**

6. Configure the authentication that the Axoflow Cloud Connector will use to access CloudWatch. Set the environment variables for the authentication method you want to use.

   - AWS Profile with a configuration file: Set the region and the AWS_PROFILE

     ```
     export AWS_PROFILE=""
     export AWS_REGION=""
     ```

   - AWS Credentials: To use AWS access keys, set an access key and a matching secret.

     ```
     export AWS_ACCESS_KEY_ID=""
     export AWS_SECRET_ACCESS_KEY=""
     export AWS_REGION=""
     ```

   - EC2 instance profile:

     ```
     export AWS_REGION=""
     ```

7. Deploy the Axoflow Cloud Connector. The exact command depends on the authentication method and the TLS settings you want to configure.

   - AWS Profile with a configuration file: Set the region and the AWS_PROFILE. Also, pass the TLS-related settings you've set earlier.

```
docker run --rm \
-v "${STORAGE_DIRECTORY}":"${STORAGE_DIRECTORY}" \
-e AWS_PROFILE="${AWS_PROFILE}" \
-e AWS_REGION="${AWS_REGION}" \
-e AWS_SDK_LOAD_CONFIG=1 \
-e AXOROUTER_ENDPOINT="${AXOROUTER_ENDPOINT}" \
-e STORAGE_DIRECTORY="${STORAGE_DIRECTORY}" \
-e AXOCLOUDCONNECTOR_DEVICE_ID="${AXOCLOUDCONNECTOR_DEVICE_ID}" \
-e <TlS-related-environment-variable>="${<TlS-related-environment-
variable>}" \
-v "${HOME}/.aws:/cloudconnectors/.aws:ro" \
ghcr.io/axoflow/axocloudconnectors:latest
```

- AWS Credentials: To use AWS access keys, set an access key and a matching secret. Also, pass the TLS-related settings you've set earlier.

```
docker run --rm \
-v "${STORAGE_DIRECTORY}":"${STORAGE_DIRECTORY}" \
-e AWS_ACCESS_KEY_ID="${AWS_ACCESS_KEY_ID}" \
-e AWS_SECRET_ACCESS_KEY="${AWS_SECRET_ACCESS_KEY}" \
-e AWS_REGION="${AWS_REGION}" \
-e AXOROUTER_ENDPOINT="${AXOROUTER_ENDPOINT}" \
-e AXOCLOUDCONNECTOR_DEVICE_ID="${AXOCLOUDCONNECTOR_DEVICE_ID}" \
-e <TlS-related-environment-variable>="${<TlS-related-environment-
variable>}" \
-e STORAGE_DIRECTORY="${STORAGE_DIRECTORY}" \
ghcr.io/axoflow/axocloudconnectors:latest
```

- EC2 instance profile: Also, pass the TLS-related settings you've set earlier.

```
docker run --rm \
-v "${STORAGE_DIRECTORY}":"${STORAGE_DIRECTORY}" \
-e AWS_REGION="${AWS_REGION}" \
-e AXOROUTER_ENDPOINT="${AXOROUTER_ENDPOINT}" \
-e AXOCLOUDCONNECTOR_DEVICE_ID="${AXOCLOUDCONNECTOR_DEVICE_ID}" \
-e <TlS-related-environment-variable>="${<TlS-related-environment-
variable>}" \
-e STORAGE_DIRECTORY="${STORAGE_DIRECTORY}" \
ghcr.io/axoflow/axocloudconnectors:latest
```

The Axoflow Cloud Connector starts forwarding logs to the AxoRouter instance.

2. Add the appliance to AxoConsole.
    1. Open the AxoConsole and select **Topology**.
    2. Select **Add Item > Source**.
    3. Select **AWS CloudWatch**.
    4. Enter the **IP address** and the **FQDN** of the Axoflow Cloud Connector instance.
    5. Select **Add**.
3. Create a Flow to route the data from the AxoRouter instance to a destination. You can use the Labels of this source to select messages from this source.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`amazon`
`product`	`meta.product`	`aws-cloudwatch`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`aws:cloudwatchlogs`	`aws-activity`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `AWS_CLOUDWATCH` .

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog` .

7.12.3 Anthropic

Send the logs of Anthropic Claude Code to Axoflow.

7.12.3.1 Claude Code

Send Anthropic Claude Code logs to Axoflow, and look up the labels that Axoflow adds to its data.

Claude Code: Agentic coding tool emitting OpenTelemetry audit events for tool use, permissions, hooks, and MCP connections.

Onboarding

Claude Code sends its events as OpenTelemetry logs. Axoflow classifies only the events that it receives on an OpenTelemetry connector: data from Claude Code received on other connectors, for example, syslog, isn't recognized as Claude Code data.

1. Create an OpenTelemetry connector rule that creates a connector on the AxoRouter you want to send the data to. By default, the connector receives data on port 4317 using the OTLP/gRPC protocol.
2. On the hosts where Claude Code runs, enable the telemetry of Claude Code and send its logs to the AxoRouter. For example, set the following environment variables:

   ```
   export CLAUDE_CODE_ENABLE_TELEMETRY=1
   export OTEL_LOGS_EXPORTER=otlp
   export OTEL_EXPORTER_OTLP_PROTOCOL=grpc
   export OTEL_EXPORTER_OTLP_ENDPOINT=https://<axorouter-address>:4317
   ```

   If the connector doesn't use TLS, use `http://` in the endpoint.

   To configure every user of your organization at once, set these variables in the managed settings of Claude Code. For details, see the Claude Code monitoring documentation.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`anthropic`
`product`	`meta.product`	`claude-code`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`anthropic:claude-code`	`infraops`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

7.12.4 Axoflow

Send logs from your AxoSyslog instances to AxoRouter using the OpenTelemetry or syslog connector.

7.12.4.1 AxoSyslog

Send data from AxoSyslog to the OpenTelemetry connector of AxoRouter, and look up the labels that Axoflow adds.

AxoSyslog: High-performance, configurable syslog service for collecting, processing, and forwarding log data.

Configure AxoSyslog to send data to an OpenTelemetry Connector of an AxoRouter instance using its `syslog-ng-otlp` destination. If that's not possible for some reason, use the `syslog-ng` destination with a Syslog Connector of an AxoRouter instance.

For the best integration of your AxoSyslog instances with AxoConsole, see AxoSyslog.

Labels

Enable classification and parsing in the connector rule that receives data from this source. Axoflow will identify the messages and add labels accordingly.

7.12.5 Balasys

Send the logs of Balasys Proxedo Network Security to Axoflow.

7.12.5.1 Proxedo Network Security

Send Balasys Proxedo Network Security logs to Axoflow, and look up the labels that Axoflow adds to its data.

Proxedo Network Security: Application-level proxy firewall suite that granularly inspects and controls network traffic at the protocol level.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	value
`vendor`	`meta.vendor`	`balasys`
`product`	`meta.product`	`pns`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`balasys:pns`	`netfw`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Earlier name/vendor

- Zorp

7.12.6 Broadcom

Send logs from Broadcom Edge SWG, Email Security, SSL Visibility Appliance, and VMware ESX, NSX, and vCenter to Axoflow.

7.12.6.1 Edge Secure Web Gateway (Edge SWG)

Send Broadcom Edge Secure Web Gateway (Edge SWG) logs to Axoflow, and look up the labels that Axoflow adds to its data.

Edge Secure Web Gateway (Edge SWG): Secures web traffic through policy enforcement, SSL inspection, and real-time threat protection.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`broadcom`
`product`	`meta.product`	`edge-swg`
`service`	`meta.service.name`	`bluecoat` , `ProxySG`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index

`bluecoat:proxysg:access:syslog`	`netops`
`bluecoat:proxysg:access:kv`	`netproxy`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `BROADCOM_EDGE_SWG` .

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog` .

Earlier name/vendor

- Blue Coat Proxy
- Blue Coat ProxySG
- Symantec ProxySG
- Symantec Edge Secure Web Gateway
- Symantec Edge SWG

7.12.6.2 Email Security

Send Broadcom Email Security logs to Axoflow, and look up the labels that Axoflow adds to its data.

Email Security: Broadcom Email Security is an enterprise-grade, high-accuracy email filtering solution designed to detect and block spam, phishing, and malware.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`broadcom`
`product`	`meta.product`	`email-security`
`service`	`meta.service.name`	`bmserver` , `ecelerity`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

source	sourcetype	index
`bmserver`	`symantec:smg`	`email`
`ecelerity`	`symantec:smg:ecelerity`	`email`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

Earlier name/vendor

- Symantec Brightmail
- Brightmail

7.12.6.3 ESX

Send Broadcom ESX logs to Axoflow, and look up the labels that Axoflow adds to its data.

ESX: The primary function of ESX is to partition a physical server into multiple virtual machines (VMs), allowing different operating systems to run simultaneously and improving hardware utilization.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`broadcom`
`product`	`meta.product`	`esx`
`service`	`meta.service.name`	`vmk` , `lsud` , `vsan` , `iofiltervpd` , `hostd` , `cmmdstimemachine` , `vmware` , `vpxa` , `eam` , `rhttpproxy` , `sdrsInjector` , `fdm` , `esxupdate` , `healthd` , `ConfigStore` , `kmxa` , `crx-cli` , `backup.sh` , `configStoreBackup` , `Host` , `vmauthd` , `localcli` , `watchdog-vsanperfsvc` , `watchdog-iofiltervpd` , `apiForwarder` , `tmpwatch` , `.etc.init.d.vsanmgmtd` , `ComplianceManager` , `hostprofiletrace` , `vobd` , `ucs-tool-esxi-inv` , `usbarb` , `auto-backup` , `cmmdsTimeMachineDump`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

source	sourcetype	index
`vmware:esxlog:<service-name>`	`vmware:esxlog:<service-name>`	`infraops`

The service-name in the source and sourcetype is the same as the `service` label, for example, `vmware:esxlog:vmware`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `VMWARE_ESX` .

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog` .

Earlier name/vendor

- VMware ESX

7.12.6.4 NSX

Send Broadcom NSX logs to Axoflow, and look up the labels that Axoflow adds to its data.

NSX: Provides network virtualization, micro-segmentation, and security for software-defined data centers.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Configure your NSX appliances, NSX Edges, and hypervisors to send their logs to the **Syslog (autodetect and classify)** connector of an AxoRouter instance. Use either:

- The TCP protocol (port 601 when using the default connector), or
- TLS-encrypted TCP protocol (port 6514 when using the default connector)

For details on configuring NSX, see Configure Remote Logging in the NSX Administration Guide.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`broadcom`
`product`	`meta.product`	`nsx`
`service`	`meta.service.name`	`NSX` , `NSXV` , `FIREWALL-PKTLOG` , `dfwpktlogs`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`vmware:nsxlog:dfwpktlogs`	`netfw`
`vmware:nsxlog:firewall-pktlog`	`netfw`
`vmware:nsxlog:nsx`	`infraops`
`vmware:nsxlog:nsxv`	`infraops`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `VMWARE_NSX`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

Earlier name/vendor

- VMware NSX
- NSX-T Data Center

7.12.6.5 SSL Visibility Appliance

Send Broadcom SSL Visibility Appliance logs to Axoflow, and look up the labels that Axoflow adds to its data.

SSL Visibility Appliance: Broadcom's SSL Visibility (SSLV) Appliance is a dedicated network security device that decrypts, inspects, and re-encrypts SSL/TLS traffic to expose threats hidden within encrypted communications.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`broadcom`
`product`	`meta.product`	`sslva`
`service`	`meta.service.name`	`ssldata`

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`broadcom:sslva`	`netproxy`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `BROADCOM_SSL_VA`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

Earlier name/vendor

Symantec SSL Visibility Appliance

7.12.6.6 vCenter

Send Broadcom vCenter logs to Axoflow, and look up the labels that Axoflow adds to its data.

vCenter: vCenter is a centralized management platform that provides a single console for managing an entire VMware vSphere virtual infrastructure, including multiple ESXi hosts and virtual machines.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`broadcom`
`product`	`meta.product`	`vcenter`
`service`	`meta.service.name`	`vpxd` , `vws` , `stats` , `cim-diag` , `sms` , `vim` , `cis-license` , `applmgmt-audit` , `updatemgr` , `vmafdd` , `vmcad` , `vmdird` , `vmon` , `osfsd` , `wcpxsvc` , `wcpsvc` , `mbcs` , `vmcam` , `vpostgres` , `vsphere` , `vcha` , `vcenter-server`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

source	sourcetype	index
`vmware:vclog:<service-name>`	`vmware:vclog:<service-name>`	`infraops`

The service-name in the source and sourcetype is the same as the `service` label, for example, `vmware:vclog:vmon`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `VMWARE_VCENTER` .

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog` .

Earlier name/vendor

- VMware vCenter

7.12.7 Check Point

Send logs from Check Point Quantum, NGFW, IPS, SmartConsole, Threat Emulation, and other Check Point products to Axoflow.

7.12.7.1 Anti-Bot

Send Check Point Software Anti-Bot logs to Axoflow, and look up the labels that Axoflow adds to its data.

Anti-Bot: Detects and blocks botnet communications and command-and-control traffic to prevent malware infections.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`checkpoint`
`product`	`meta.product`	`anti-bot`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	source	index
`cp_log`	`checkpoint:endpoint`	`netops`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CHECKPOINT_EDR`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.7.2 Anti-Malware

Send Check Point Software Anti-Malware logs to Axoflow, and look up the labels that Axoflow adds to its data.

Anti-Malware: Protects endpoints from viruses, ransomware, and other malware using signature and behavior analysis.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`checkpoint`
`product`	`meta.product`	`anti-malware`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	source	index
`cp_log`	`checkpoint:endpoint`	`netops`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CHECKPOINT_EDR`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.7.3 Anti-Phishing

Send Check Point Software Anti-Phishing logs to Axoflow, and look up the labels that Axoflow adds to its data.

Anti-Phishing: Prevents phishing attacks by analyzing email content and links to block credential theft attempts.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`checkpoint`
`product`	`meta.product`	a `nti-phishing`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	source	index
`cp_log`	`checkpoint:email`	`email`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CHECKPOINT_EMAIL`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.7.4 Anti-Spam and Email Security

Send Check Point Software Anti-Spam and Email Security logs to Axoflow, and look up the labels that Axoflow adds to its data.

Anti-Spam and Email Security: Blocks spam and malicious email content using reputation checks and email filtering techniques.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`checkpoint`
`product`	`meta.product`	`antispam-emailsecurity`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	source	index
`cp_log`	`checkpoint:email`	`email`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CHECKPOINT_EMAIL`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.7.5 CPMI Client

Send Check Point Software CPMI Client logs to Axoflow, and look up the labels that Axoflow adds to its data.

CPMI Client: Legacy Check Point management client used to interface with security policies and logs.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`checkpoint`
`product`	`meta.product`	`cpmi-client`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`cp_log`	`netops`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CHECKPOINT_FIREWALL`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.7.6 cpmidu_update_tool

Send Check Point Software cpmidu_update_tool logs to Axoflow, and look up the labels that Axoflow adds to its data.

cpmidu_update_tool: Utility used to update configuration and database files for Check Point Multi-Domain environments.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`checkpoint`
`product`	`meta.product`	`cpmidu-update-tool`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	source	index
`cp_log`	`checkpoint:audit`	`netops`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CHECKPOINT_FIREWALL`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.7.7 Database Tool

Send Check Point Software Database Tool logs to Axoflow, and look up the labels that Axoflow adds to its data.

Database Tool: Command-line tool to extract, query, or update Check Point configuration and policy databases.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`checkpoint`
`product`	`meta.product`	`database-tool`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	source	index
`cp_log`	`checkpoint:audit`	`netops`

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.7.8 Endpoint Compliance

Send Check Point Software Endpoint Compliance logs to Axoflow, and look up the labels that Axoflow adds to its data.

Endpoint Compliance: Checks endpoint status and posture before granting network access, enforcing security policies.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`checkpoint`
`product`	`meta.product`	`endpoint-compliance`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	source	index
`cp_log`	`checkpoint:endpoint`	`netops`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CHECKPOINT_EDR`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.7.9 Endpoint Management

Send Check Point Software Endpoint Management logs to Axoflow, and look up the labels that Axoflow adds to its data.

Endpoint Management: Centralized platform for managing endpoint protection, updates, and policy enforcement.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`checkpoint`
`product`	`meta.product`	`endpoint-management`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	source	index
`cp_log`	`checkpoint:endpoint`	`netops`

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CHECKPOINT_EDR`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.7.10 Forensics

Send Check Point Software Forensics logs to Axoflow, and look up the labels that Axoflow adds to its data.

Forensics: Analyzes security incidents on endpoints to uncover attack vectors and malicious activity.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`checkpoint`
`product`	`meta.product`	`forensics`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	source	index
`cp_log`	`checkpoint:endpoint`	`netops`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CHECKPOINT_EDR`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.7.11 GO Password Reset

Send Check Point Software GO Password Reset logs to Axoflow, and look up the labels that Axoflow adds to its data.

GO Password Reset: Facilitates secure password reset processes for users across integrated environments.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`checkpoint`
`product`	`meta.product`	`go-password-reset`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	source	index
`cp_log`	`checkpoint:audit`	`netops`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CHECKPOINT_AUDIT`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.7.12 HTTPS Inspection

Send Check Point Software HTTPS Inspection logs to Axoflow, and look up the labels that Axoflow adds to its data.

HTTPS Inspection: Decrypts and inspects HTTPS traffic to detect hidden threats within encrypted web sessions.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`checkpoint`
`product`	`meta.product`	`https-inspection`
`service`	`meta.service.name`	`CP-GW`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	source	index
`cp_log`	`checkpoint:firewall`	`netfw`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the

name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CHECKPOINT_FIREWALL`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.7.13 iOS Profiles

Send Check Point Software iOS Profiles logs to Axoflow, and look up the labels that Axoflow adds to its data.

iOS Profiles: Provides configuration profiles for secure mobile access and web filtering on iOS devices.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`checkpoint`
`product`	`meta.product`	`ios-profiles`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	source	index
`cp_log`	`checkpoint:network`	`netops`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CHECKPOINT_HARMONY`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.7.14 IPS

Send Check Point Software IPS logs to Axoflow, and look up the labels that Axoflow adds to its data.

IPS: Detects and blocks known and unknown exploits, malware, and vulnerabilities in network traffic.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`checkpoint`
`product`	`meta.product`	`ips`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	source	index
`cp_log`	`checkpoint:ids`	`netids`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CHECKPOINT_EDR`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.7.15 MDS Query Tool

Send Check Point Software MDS Query Tool logs to Axoflow, and look up the labels that Axoflow adds to its data.

MDS Query Tool: CLI tool for querying multi-domain configurations and policies in Check Point environments.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`checkpoint`
`product`	`meta.product`	`mds-query-tool`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`cp_log`	`netops`

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.7.16 Media Encryption & Port Protection

Send Check Point Software Media Encryption & Port Protection logs to Axoflow, and look up the labels that Axoflow adds to its data.

Media Encryption & Port Protection: Secures USB ports and encrypts removable media to protect sensitive data on endpoints.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`checkpoint`
`product`	`meta.product`	`media-port`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	source	index
`cp_log`	`checkpoint:endpoint`	`netops`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CHECKPOINT_EDR`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.7.17 Mobile Access

Send Check Point Software Mobile Access logs to Axoflow, and look up the labels that Axoflow adds to its data.

Mobile Access: Enables secure remote access to corporate apps and data from mobile devices.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`checkpoint`
`product`	`meta.product`	`mobile-access`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	source	index
`cp_log`	`checkpoint:network`	`netops`

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.7.18 Next-Generation Firewall (NGFW)

Send Check Point Software Next-Generation Firewall (NGFW) logs to Axoflow, and look up the labels that Axoflow adds to its data.

Next-Generation Firewall (NGFW): Next-generation firewall providing intrusion prevention, application control, and threat protection.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`checkpoint`
`product`	`meta.product`	`firewall`
`service`	`meta.service.name`	`CP-GW`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	source	index
`cp_log`	`checkpoint:firewall`	`netfw`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the

name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CHECKPOINT_FIREWALL`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.7.19 QoS

Send Check Point Software QoS logs to Axoflow, and look up the labels that Axoflow adds to its data.

QoS: Implements bandwidth control and traffic prioritization policies for optimized network usage.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`checkpoint`
`product`	`meta.product`	`qos`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	source	index
`cp_log`	`checkpoint:firewall`	`netfw`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CHECKPOINT_FIREWALL`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.7.20 Quantum

Support for Check Point Quantum is coming soon. Request it, or ask about the status of the integration.

Quantum: Unified threat prevention platform delivering firewall, VPN, and intrusion prevention capabilities.

If you'd like to send data from this source to AxoRouter, contact our support team for details.

7.12.7.21 Query Database

Send Check Point Software Query Database logs to Axoflow, and look up the labels that Axoflow adds to its data.

Query Database: Accesses and queries internal policy or object databases in Check Point systems.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`checkpoint`
`product`	`meta.product`	`query-database`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	source	index
`cp_log`	`checkpoint:audit`	`netops`

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.7.22 SmartConsole

Send Check Point Software SmartConsole logs to Axoflow, and look up the labels that Axoflow adds to its data.

SmartConsole: Graphical interface for managing Check Point security policies, logs, and monitoring.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`checkpoint`
`product`	`meta.product`	`smartconsole`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	source	index
`cp_log`	`checkpoint:audit`	`netops`

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.7.23 SmartUpdate

Send Check Point Software SmartUpdate logs to Axoflow, and look up the labels that Axoflow adds to its data.

SmartUpdate: Tool for updating and managing licenses, software, and hotfixes in Check Point environments.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`checkpoint`
`product`	`meta.product`	`smartupdate`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	source	index
`cp_log`	`checkpoint:audit`	`netops`

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.7.24 Syslog Data

Send Check Point Software Syslog Data logs to Axoflow, and look up the labels that Axoflow adds to its data.

Syslog Data: Enables export of Check Point logs via syslog to external monitoring and SIEM platforms.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`checkpoint`
`product`	`meta.product`	`syslog`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`nix:syslog`	`netops`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.7.25 Threat Emulation and Anti-Exploit

Send Check Point Software Threat Emulation and Anti-Exploit logs to Axoflow, and look up the labels that Axoflow adds to its data.

Threat Emulation and Anti-Exploit: Emulates files in a virtual environment to detect and block advanced persistent threats and exploits.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`checkpoint`
`product`	`meta.product`	`threat-emulation`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	source	index
`cp_log`	`checkpoint:endpoint`	`netops`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CHECKPOINT_EDR`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.7.26 URL Filtering

Send Check Point Software URL Filtering logs to Axoflow, and look up the labels that Axoflow adds to its data.

URL Filtering: Controls and logs web access based on URL categories and custom site rules to enforce policy.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`checkpoint`
`product`	`meta.product`	`url-filtering`
`service`	`meta.service.name`	`CP-GW`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	source	index
`cp_log`	`checkpoint:firewall`	`netfw`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the

name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CHECKPOINT_FIREWALL`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.7.27 Web API

Send Check Point Software Web API logs to Axoflow, and look up the labels that Axoflow adds to its data.

Web API: Provides programmatic access to Check Point security management through RESTful API endpoints.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`checkpoint`
`product`	`meta.product`	`web-api`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	source	index
`cp_log`	`checkpoint:audit`	`netops`

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.8 Cisco

Send logs from Cisco ASA, Firepower, FTD, IOS, ISE, Meraki, and other Cisco devices to Axoflow.

7.12.8.1 Access Control System (ACS)

Send Cisco Access Control System (ACS) logs to Axoflow, and look up the labels that Axoflow adds to its data.

Access Control System (ACS): Centralizes network access control with RADIUS and TACACS+ for authentication and authorization.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`cisco`
`product`	`meta.product`	`acs`
`service`	`meta.service.name`	`CSCOacs_Single_Authentications`, `CSCOacs_Multi_Authentications`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`cisco:acs`	`netauth`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CISCO_ACS`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.8.2 Adaptive Security Appliance (ASA)

Send Cisco Adaptive Security Appliance (ASA) logs to Axoflow, and look up the labels that Axoflow adds to its data.

Adaptive Security Appliance (ASA): Provides stateful firewall, VPN support, and advanced threat protection for secure network perimeters.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`cisco`
`product`	`meta.product`	`asa`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`cisco:asa`	`netfw`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CISCO_ASA_FIREWALL`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.8.3 Application Control Engine (ACE)

Send Cisco Application Control Engine (ACE) logs to Axoflow, and look up the labels that Axoflow adds to its data.

Application Control Engine (ACE): Provides application-aware load balancing, SSL offload, and traffic control for Cisco networks.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`cisco`
`product`	`meta.product`	`ace`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`cisco:ace`	`netops`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CISCO_ACE`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.8.4 Cisco IOS

Send Cisco Internetwork Operating System logs to Axoflow, and look up the labels that Axoflow adds to its data.

Cisco IOS: Network operating system for Cisco routers and switches, enabling routing, switching, and security.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`cisco`
`product`	`meta.product`	`ios`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`cisco:ios`	`netops`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CISCO_IOS`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.8.5 Digital Network Architecture (DNA)

Send Cisco Digital Network Architecture (DNA) logs to Axoflow, and look up the labels that Axoflow adds to its data.

Digital Network Architecture (DNA): Provides software-defined networking, policy automation, and analytics for enterprise infrastructure.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`cisco`
`product`	`meta.product`	`dna`
`service`	`meta.service.name`	`DNAC`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`cisco:dna`	`netops`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the

name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CISCO_DNAC`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.8.6 Email Security Appliance (ESA)

Send Cisco Email Security Appliance (ESA) logs to Axoflow, and look up the labels that Axoflow adds to its data.

Email Security Appliance (ESA): Protects email systems from spam, phishing, malware, and data loss with advanced threat filtering.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Note that the device can be configured to send plain-text syslog or CEF-formatted logs. AxoRouter can automatically parse all flavors.

Labels

Axoflow automatically adds the following labels to data collected from this source:

label	Value
`vendor`	`meta.vendor`
`product`	`meta.product`
`service`	`meta.service.name`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype, index, and source settings:

sourcetype	index	source
cisco:esa:http	email	esa:http
cisco:esa:textmail	email	esa:textmail

cisco:esa:amp	email	esa:amp
cisco:esa:antispam	email	esa:antispam
cisco:esa:system_logs	email	esa:system_logs
cisco:esa:system_logs	email	esa:euq_logs
cisco:esa:system_logs	email	esa:service_logs
cisco:esa:system_logs	email	esa:reportd_logs
cisco:esa:system_logs	email	esa:sntpd_logs
cisco:esa:system_logs	email	esa:smartlicense
cisco:esa:error_logs	email	esa:error_logs
cisco:esa:error_logs	email	esa:updater_logs
cisco:esa:content_scanner	email	esa:content_scanner
cisco:esa:authentication	email	esa:authentication
cisco:esa:http	email	esa:http
cisco:esa:textmail	email	esa:textmail
cisco:esa:amp	email	esa:amp
cisco:esa	email	program: <variable>
cisco:esa:cef	email	esa:consolidated

Tested with: Splunk Add-on for Cisco ESA

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CISCO_EMAIL_SECURITY`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.8.7 Firepower

Send Cisco Firepower logs to Axoflow, and look up the labels that Axoflow adds to its data.

Firepower: Provides next-gen firewall features including intrusion prevention, app control, and malware protection.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`cisco`
`product`	`meta.product`	`firepower`
`service`	`meta.service.name`	`SFIMS`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`cisco:firepower:syslog`	`netids`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the

name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CISCO_FIREPOWER_FIREWALL`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.8.8 Firepower Threat Defence (FTD)

Send Cisco Firepower Threat Defence (FTD) logs to Axoflow, and look up the labels that Axoflow adds to its data.

Firepower Threat Defence (FTD): Unifies firewall, VPN, and intrusion prevention into a single software for comprehensive threat defense.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`cisco`
`product`	`meta.product`	`ftd`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`cisco:ftd`	`netfw`

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CISCO_FIREPOWER_FIREWALL`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.8.9 Firewall Services Module (FWSM)

Send Cisco Firewall Services Module (FWSM) logs to Axoflow, and look up the labels that Axoflow adds to its data.

Firewall Services Module (FWSM): Delivers multi-context, high-performance firewall services integrated into Cisco Catalyst switches.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`cisco`
`product`	`meta.product`	`fwsm`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`cisco:fwsm`	`netfw`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CISCO_FWSM`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.8.10 HyperFlex (HX, UCSH)

Send Cisco HyperFlex (HX, UCSH) logs to Axoflow, and look up the labels that Axoflow adds to its data.

HyperFlex (HX, UCSH): Infrastructure solution combining compute, storage, and networking in a single system.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`cisco`
`product`	`meta.product`	`ucsh`
`service`	`meta.service.name`	`hx-audit-rest` , `hx-device-connector` , `hx-ssl-access`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`cisco:ucsh:hx`	`infraops`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the

name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CISCO_UCS`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.8.11 Identity Services Engine (ISE)

Send Cisco Identity Services Engine (ISE) logs to Axoflow, and look up the labels that Axoflow adds to its data.

Identity Services Engine (ISE): Manages network access control and enforces policies with user and device authentication capabilities.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

For details on configuring your Identity Services Engine to forward its logs to an AxoRouter instance, see Configure Remote Syslog Collection Locations in Cisco Identity Services Engine (ISE) Administrator Guide.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`cisco`
`product`	`meta.product`	`ise`
`service`	`meta.service.name`	`CISE_Alarm` , `CISE_Passed_Authentications` , `CISE_RADIUS_Accounting` , `CISE_System_Statistics`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index

`cisco:ise:syslog`	`netauth`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CISCO_ISE` .

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog` .

7.12.8.12 Integrated Management Controller (IMC)

Send Cisco Integrated Management Controller (IMC) logs to Axoflow, and look up the labels that Axoflow adds to its data.

Integrated Management Controller (IMC): Provides out-of-band server management for Cisco UCS, enabling hardware monitoring and configuration.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`cisco`
`product`	`meta.product`	`cimc`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`cisco:cimc`	`infraops`

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CISCO_UCS`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.8.13 IOS XR

Send Cisco IOS XR logs to Axoflow, and look up the labels that Axoflow adds to its data.

IOS XR: High-performance, modular network operating system for carrier-grade routing and scalability.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`cisco`
`product`	`meta.product`	`xr`
`service`	`meta.service.name`	`config` , `nfsvr` , `plat_sl_client` , `ssh_syslog_proxy`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`cisco:xr`	`netops`

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CISCO_ROUTER`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.8.14 Meeting Management

Send Cisco Meeting Management logs to Axoflow, and look up the labels that Axoflow adds to its data.

Meeting Management: Cisco Meeting Management is a management tool for Cisco Meeting Server, an on-premises video meeting platform.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`cisco`
`product`	`meta.product`	`mm`
`service`	`meta.service.name`	Service name detected in the message, for example, `cmm-web_events_monitor_1`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

source	sourcetype	index
`cmm-<service-name>`	`cisco:mm:system`	`netops`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CISCO_CALL_MANAGER`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.8.15 Meraki MX

Send Cisco Meraki MX logs to Axoflow, and look up the labels that Axoflow adds to its data.

Meraki MX: Cloud-managed network appliance offering firewall, VPN, SD-WAN, and security in a single platform.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`cisco`
`product`	`meta.product`	`meraki`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`cisco:meraki`	`netfw`

Tested with: TA-meraki

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the

name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CISCO_MERAKI`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.8.16 Private Internet eXchange (PIX)

Send Cisco Private Internet eXchange (PIX) logs to Axoflow, and look up the labels that Axoflow adds to its data.

Private Internet eXchange (PIX): Legacy firewall appliance delivering stateful inspection and secure network access control.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`cisco`
`product`	`meta.product`	`pix`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`cisco:pix`	`netfw`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CISCO_PIX_FIREWALL`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.8.17 TelePresence Video Communication Server (VCS)

Send Cisco TelePresence Video Communication Server (VCS) logs to Axoflow, and look up the labels that Axoflow adds to its data.

TelePresence Video Communication Server (VCS): Enables video conferencing control and call routing for Cisco TelePresence systems and endpoints.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`cisco`
`product`	`meta.product`	`tvcs`
`service`	`meta.service.name`	`tvcs`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`cisco:tvcs`	`main`

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.8.18 Unified Computing System Manager (UCSM)

Send Cisco Unified Computing System Manager (UCSM) logs to Axoflow, and look up the labels that Axoflow adds to its data.

Unified Computing System Manager (UCSM): Centralized management platform for Cisco Unified Computing System (UCS) servers and resources.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`cisco`
`product`	`meta.product`	`ucsm`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`cisco:ucs`	`infraops`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CISCO_UCS`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.8.19 Unified Communications Manager (UCM)

Send Cisco Unified Communications Manager (UCM) logs to Axoflow, and look up the labels that Axoflow adds to its data.

Unified Communications Manager (UCM): Delivers unified voice, video, messaging, and mobility services in enterprise IP telephony systems.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`cisco`
`product`	`meta.product`	`ucm`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`cisco:ucm`	`netops`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CISCO_UCM`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.8.20 Viptela

Send Cisco Viptela logs to Axoflow, and look up the labels that Axoflow adds to its data.

Viptela: Software-defined WAN solution providing secure connectivity, centralized control, and traffic optimization.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`cisco`
`product`	`meta.product`	`viptela`
`service`	`meta.service.name`	`SYSMGR`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`cisco:viptela`	`netops`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the

name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CISCO_VIPTELA`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.9 Citrix

Send logs from Citrix NetScaler application delivery controllers to Axoflow.

7.12.9.1 Netscaler

Send Citrix NetScaler logs to Axoflow, and look up the labels that Axoflow adds to its data.

Netscaler: Offers application delivery, load balancing, and security features for optimized app performance.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`citrix`
`product`	`meta.product`	`netscaler`
`service`	`meta.service.name`	`svm_service`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index

`citrix:netscaler:appfw:cef`	`netfw`
`citrix:netscaler:syslog`	`netfw`
`citrix:netscaler:appfw`	`netfw`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CITRIX_NETSCALER_WEB_LOGS` .

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog` .

7.12.10 Cohesity

7.12.10.1 DataProtect

Send Cohesity DataProtect audit and data protection logs to Axoflow, and look up the labels that Axoflow adds to its data.

DataProtect: Backup and recovery cluster that sends cluster audit, API access, data protection, and SMB file audit events.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	value
`vendor`	`meta.vendor`	`cohesity`
`product`	`meta.product`	`dataprotect`
`service`	`meta.service.name`	`cluster_audit`, `api_audit`, `dataprotection_events`, `filesystem_audit`, `shell_commands_audit`, or `cli_audit`

The `service` label depends on the Cohesity log that the message comes from (the program name in the syslog header):

- `cluster_audit`: Cluster audit log
- `api_audit`: API access audit log
- `dataprotection_events`: Data protection events
- `filesystem_audit`: File system (SMB) audit log
- `shell_commands_audit`: Shell commands audit log
- `cli_audit`: CLI audit log

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`cohesity:api:audit`	`infraops`
`cohesity:cluster:audit`	`infraops`
`cohesity:cluster:dataprotection`	`infraops`

The `api_audit` service uses the `cohesity:api:audit` sourcetype, the `dataprotection_events` service uses `cohesity:cluster:dataprotection`, and every other service uses `cohesity:cluster:audit`.

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `COHESITY`.

7.12.11 Corelight

Forward logs from the Corelight Open Network Detection and Response (NDR) platform to Axoflow.

7.12.11.1 Open Network Detection & Response (NDR)

Send Corelight NDR Platform logs to Axoflow, and look up the labels that Axoflow adds to its data.

Open Network Detection & Response (NDR): Provides network detection and response by analyzing traffic for advanced threats and anomalous behavior.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`corelight`
`product`	`meta.product`	`ndr-platform`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype, source, and index settings:

sourcetype	index
`corelight_alerts`	`main`

corelight_conn	main
corelight_corelight	main
corelight_corelight_metrics_bro	main
corelight_corelight_metrics_iface	main
corelight_dhcp	main
corelight_dpd	main
corelight_etc_viz	main
corelight_evt_all	main
corelight_evt_http	main
corelight_evt_suri	main
corelight_files	main
corelight_ftp	main
corelight_http	main
corelight_http_red	main
corelight_idx	main
corelight_irc	main
corelight_kerberos	main
corelight_metrics_bro	main
corelight_metrics_iface	main
corelight_rdp	main
corelight_smb	main
corelight_smb_files	main
corelight_socks	main
corelight_ssh	main
corelight_ssh_red	main
corelight_ssl	main

`corelight_st_base`	`main`
`corelight_suri`	`main`
`corelight_suricata_corelight`	`main`
`corelight_x509`	`main`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CORELIGHT` .

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog` .

7.12.12 CrowdStrike

Collect security events from the CrowdStrike Falcon platform and send them to Axoflow.

7.12.12.1 CrowdStrike Falcon Platform

Collect CrowdStrike Falcon data with a Cloud Connector, or receive CEF data from Falcon over syslog, and look up the labels that Axoflow adds.

CrowdStrike Falcon Platform: CrowdStrike Falcon is a cloud-native, AI-powered cybersecurity platform designed to stop breaches by protecting endpoints, cloud workloads, identity, and data.

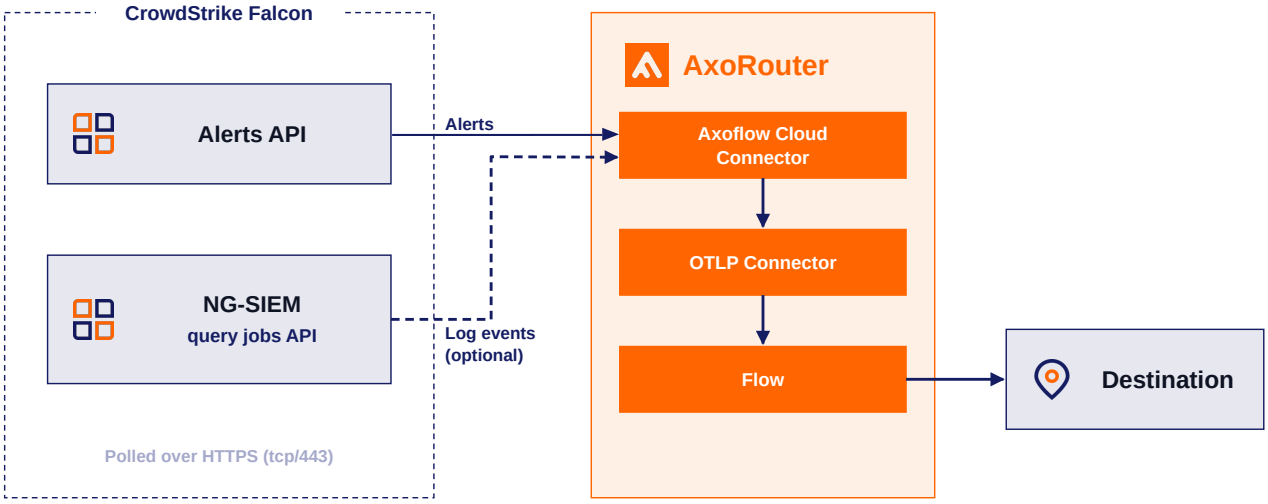
> **Note**
>
> **Axoflow can use CrowdStrike Falcon as a source and as a destination. To configure a CrowdStrike Falcon destination, see CrowdStrike.**

Axoflow can collect data from Falcon in two ways:

- With an Axoflow Cloud Connector that polls the Falcon APIs for alerts, and optionally runs searches on an NG-SIEM repository.
- By receiving CEF messages that Falcon sends over syslog.

Collect data with Axoflow Cloud Connector

- Create a **Connector Rule** on the AxoRouter that deploys an Axoflow Cloud Connector and collects the data from the Falcon APIs. Axoflow Cloud Connector is a simple container that runs on an AxoRouter instance.
- Configure a Flow on AxoConsole that processes and routes the collected data to your destination (for example, Splunk or another SIEM).



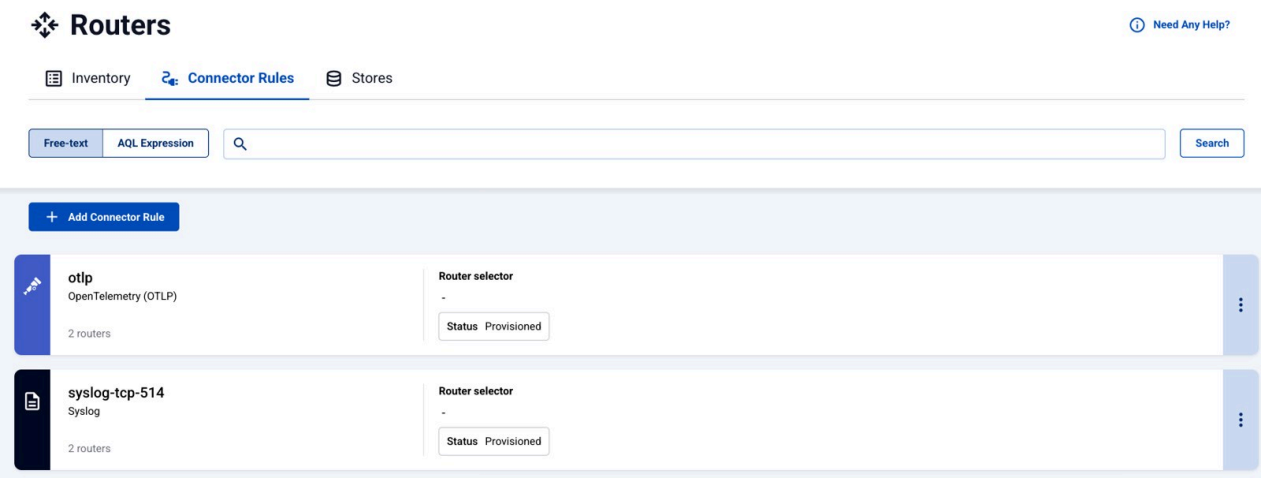
Prerequisites

- An AxoRouter instance that:
  - Was provisioned with **Enable Cloudconnectors** turned on.
  - Can reach the API host of your Falcon cloud (for example, api.crowdstrike.com) on TCP port 443. Adjust your firewall rules and the access rules of the data source as needed.
  - You know the IP address the AxoRouter. To find it:
    1. Open the AxoConsole.
    2. Select the **Routers** or the **Topology** page.
    3. Select on AxoRouter instance that is going to receive the logs.
    4. Check the **Networks > Address** field.
- A CrowdStrike Falcon subscription.
- A Falcon API client, and the client ID and client secret of this client. The API client must be able to read the Alerts API. To run NG-SIEM searches, it must also be able to run query jobs.

Configure Axoflow Cloud Connector on AxoRouter

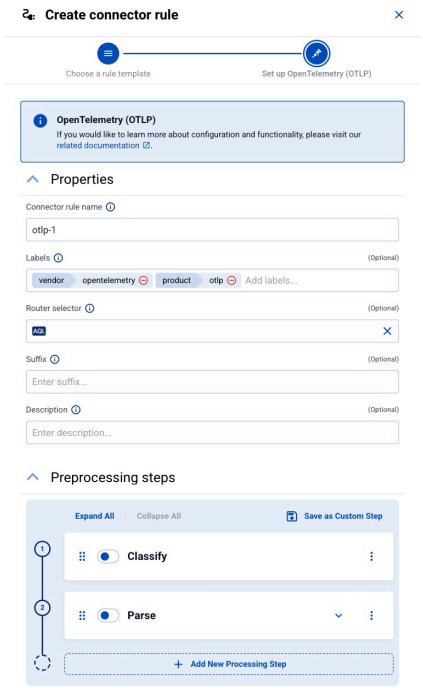
To collect data from CrowdStrike Falcon, complete the following steps.

1. Select **Routers > Connector Rules > Add Connector Rule**. (Alternatively, you can select **Add Connector > Add a connector rule** on the **Connectors** page of an AxoRouter host.)



2. Select the **Falcon Platform**.

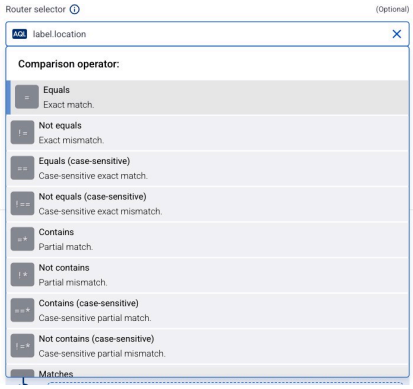
3. Enter a name for the connector rule into the **Connector rule name** field.



4. (Optional) Add labels to the connector rule. You will be able to use these labels in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps. For details about the message schema and the related connector-specific fields, see the `meta.connector` object.

5. Set the **Router selector** for the connector rule. The selector determines which AxoRouter instances get a connector from this connector rule.

> **CAUTION:**
>
> Each connector rule for CrowdStrike Falcon must match exactly one AxoRouter. Use the `name` or `guid` AQL selector to make sure it matches only one AxoRouter. Otherwise several Axoflow Cloud Connector instances collect the same data, and Axoflow receives every message multiple times.



You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

- If you leave the router selector empty, it matches every AxoRouter instance.
- To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.
- If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

6. (Optional) Enter a **Suffix** for the connector rule. AxoConsole uses the suffix in the name of the connector instances that it creates on the AxoRouter hosts. The name of the connector instance follows the `<name-of-AxoRouter>-<suffix>` schema. For example, if the name of a matching AxoRouter instance is `axorouter-1`, and the suffix of the rule is `custom`, the name of the connector on that AxoRouter is `axorouter-1-custom`.

   If the **Suffix** field is empty, AxoConsole uses the name of the connector rule instead. For example, if the name of the connector rule is `otlp-1`, the name of the connector is `axorouter-1-otlp-1`.

7. (Optional) Enter a description for the rule.

8. Configure how Axoflow Cloud Connector reaches the Falcon APIs.

   - **Client ID** and **Client secret**: The credentials of the Falcon API client that Axoflow Cloud Connector uses to poll the platform.
   - **Cloud**: The Falcon cloud to poll: `us-1`, `us-2`, `us-3`, `eu-1`, `us-gov-1`, `us-gov-2`, `gov-1`, or `gov-2`. Select **Autodiscover** to resolve the cloud from the credentials of the API client.
   - **Member CID**: (Optional) Selects a CID if the API client has access to several of them.

9. (Optional) Select **More options** to change the advanced options of the connector, for example, to collect data from an NG-SIEM repository.

10. Select **Add**. The Axoflow Cloud Connector starts polling the Falcon APIs and forwarding the collected events to the AxoRouter instance.
11. Verify that the Axoflow Cloud Connector is running.
    1. Select **Routers**, then select the AxoRouter you added the Axoflow Cloud Connector to.
    2. Select **Services** and check that **axoflow-cloudconnector.service** is running without errors.
12. Create a Flow to route the data from the AxoRouter instance to a destination. You can use the Labels of this source to select its messages.

Advanced options

Select **More options** on the connector rule to set the following options.

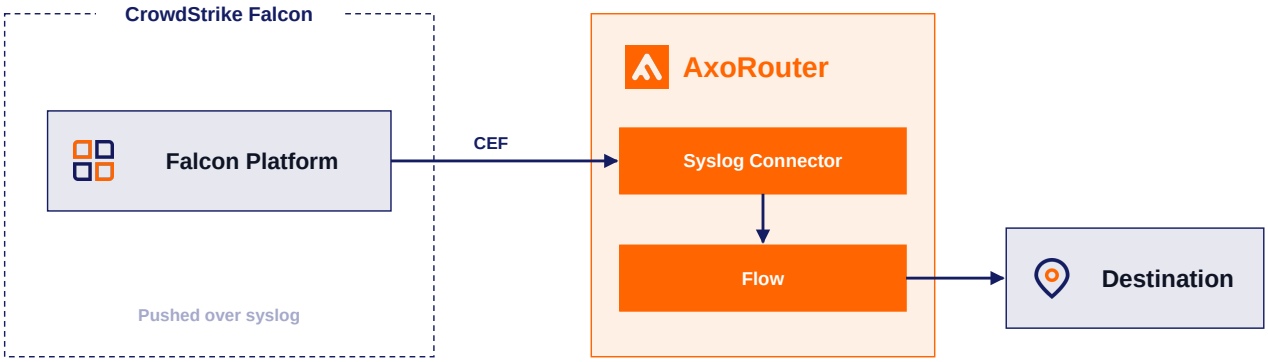
- **Host override**: Poll this host instead of the one that the selected cloud resolves to.
- **Poll interval**: How often Axoflow Cloud Connector polls the Falcon APIs. Default value: `30s`
- **Initial lookback**: How far back the first poll reaches when no checkpoint is stored yet. Set it to `0` to collect only the events that arrive after the connector starts.

  Axoflow Cloud Connector checkpoints the position of the poll cursors, so after a restart it resumes the collection where the last poll stopped. The **Initial lookback** option applies only when there is no stored checkpoint.
- **Disable alerts poller**: Don't fetch alerts. If you enable this option, enable the **NG-SIEM search** option as well. Otherwise, there's nothing to collect, and the connector rejects the configuration.
- **NG-SIEM search**: Collects log events from an NG-SIEM repository using the query jobs API. Enable it to set the following options.
  - **Repository**: The NG-SIEM repository (view) to query.
  - **Query string**: The CQL filter that selects the events to collect. By default, the connector collects all events. Every matching event becomes a separate log record, so don't use aggregating functions in the query.
  - **Search poll interval**: Overrides **Poll interval** for the search poller, because a query job is much more expensive than a page of alerts. If you leave it empty, the search poller uses the **Poll interval** value.

Receive CEF data from Falcon

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Configure Falcon to send data in CEF (Common Event Format) to Axoflow.



Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`crowdstrike`
`product`	`meta.product`	`falcon`
`service`	`meta.service.name`	The value of the `deviceProcessName` field of the log message, for example, `Crowdstrike Streaming API`. Axoflow sets this label only for CEF messages received over syslog.

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When you send the CEF data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`crowdstrike:detections:cef` , `crowdstrike:event:cef` , `crowdstrike:source_stream:cef`	`default`

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CS_STREAM` .

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `CommonSecurityLog` .

7.12.13 CyberArk

Send logs from CyberArk Vault and Privileged Threat Analytics (PTA) to Axoflow.

7.12.13.1 Privileged Threat Analytics (PTA)

Send CyberArk Privileged Threat Analytics (PTA) logs to Axoflow, and look up the labels that Axoflow adds to its data.

Privileged Threat Analytics (PTA): Analyzes privileged account behavior to detect threats and suspicious activity in real time.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`cyberark`
`product`	`meta.product`	`pta`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`cyberark:pta:cef`	`main`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CYBERARK_PTA`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.13.2 Vault

Send CyberArk Vault logs to Axoflow, and look up the labels that Axoflow adds to its data.

Vault: Stores and manages privileged credentials, session recordings, and access control policies securely.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`cyberark`
`product`	`meta.product`	`vault`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`cyberark:epv:cef`	`netauth`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `CYBERARK`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.14 Elastic

Collect data from Elasticsearch, or receive data from Beats, Elastic Agent, and Logstash.

7.12.14.1 Beats, Elastic Agent, Logstash

Receive data from Beats, Elastic Agent, and Logstash with the Elasticsearch Bulk API connector of AxoRouter.

Axoflow can receive data directly from clients like Beats, Logstash, or Elastic Agent using its Elasticsearch Bulk API implementation. For details, see the Elasticsearch Bulk API connector.

7.12.14.2 Elasticsearch

Collect data that's already stored in Elasticsearch indexes with a Cloud Connector, and look up the labels Axoflow adds.

Axoflow can collect data from Elasticsearch using an Axoflow Cloud Connector. At a high level, the process looks like this:

- Deploy an Axoflow Cloud Connector that will collect the data from Elastic. Axoflow Cloud Connector is a simple container that you can deploy into your cloud provider, or on-prem.
- The connector polls the Elasticsearch `_search` API and forwards the collected data to the OpenTelemetry connector of an AxoRouter instance. This AxoRouter can be deployed within the cloud, or on-prem. You can deploy Axoflow Cloud Connector on the same host as AxoRouter.
- Configure a Flow on AxoConsole that processes and routes the collected data to your destination (for example, Splunk or another SIEM).

> **Note**
>
> **This connector collects data that's already stored in an Elasticsearch index by polling the Elasticsearch `_search` API. To receive data directly from clients (like Beats, Logstash, or Elastic Agent), use the Elasticsearch Bulk API connector.**

Prerequisites

- An Elastic Cloud or Elastic Stack account with an active subscription.
- A virtual machine or Kubernetes node to deploy Axoflow Cloud Connector on. This can be the same host that'll run AxoRouter.
- An AxoRouter instance that can receive data from the connector. Verify that it has an OpenTelemetry Connector (it's enabled by default). This can be the same host that'll run Axoflow Cloud Connector.
- You know the IP address the AxoRouter. To find it:

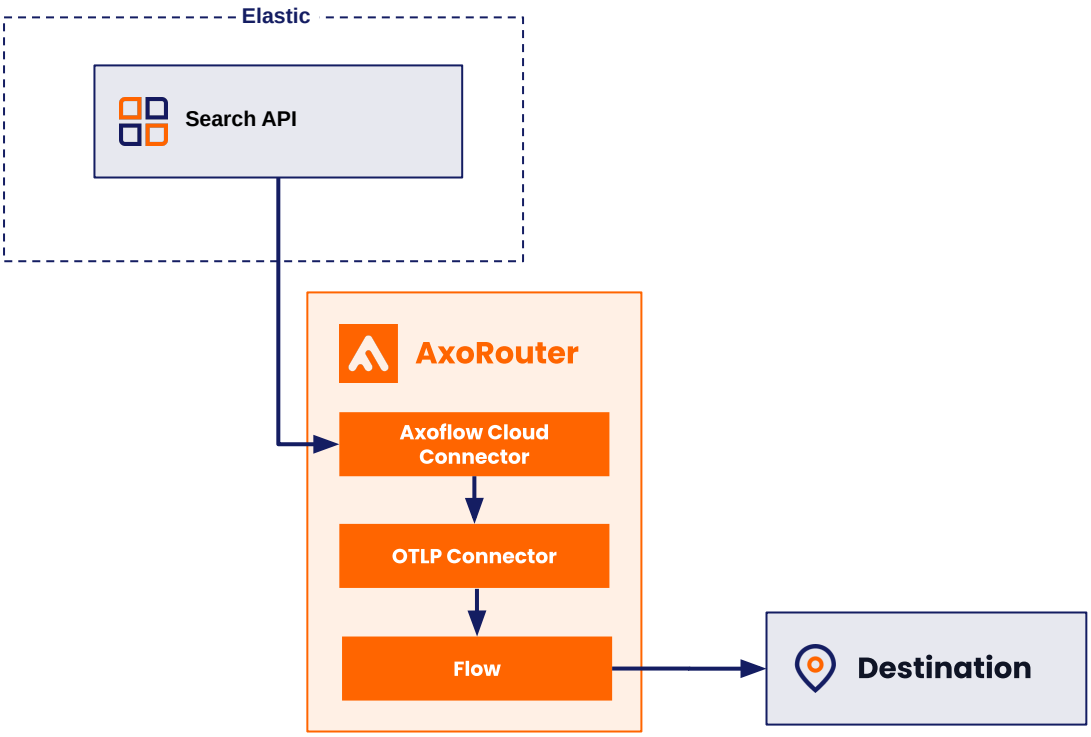
  1. Open the AxoConsole.
  2. Select the **Routers** or the **Topology** page.
  3. Select on AxoRouter instance that is going to receive the logs.
  4. Check the **Networks > Address** field.
- The Axoflow Cloud Connector must be able to access the AxoRouter on the port the OpenTelemetry Connector is listening on (by default, port 4317). Depending on where the Axoflow Cloud Connector and AxoRouter are deployed, you might need to adjust

firewall and ingress/egress rules in your environment.

- API key for Axoflow Cloud Connector to authenticate to your Elastic instance.

Axoflow Cloud Connector and AxoRouter on the same host

To collect data from Elastic when Axoflow Cloud Connector and AxoRouter are running on the same host, complete the following steps.



1. Deploy AxoRouter on a Linux virtual machine: select **Provisioning > AxoRouter > Linux**, then run the one-liner provisioning command on the host.

2. Download the Axoflow Cloud Connector image. The host must be able to access `ghcr.io` on TCP port 443 for this step.

```
sudo podman pull ghcr.io/axoflow/axocloudconnectors:latest
```

3. Ensure that Axoflow Cloud Connector can access the Elastic APIs. (For Elastic Cloud, Axoflow Cloud Connector uses HTTPS over port 443).

4. Install the Axoflow Cloud Connector.

    1. Create the `/etc/axorouter/cloudconnectors-elastic.env` file with the following content:

```
AXOCLOUDCONNECTORS_UID=0
AXOCLOUDCONNECTORS_GID=0
AXOCLOUDCONNECTORS_CAPS=CAP_NET_BIND_SERVICE CAP_NET_BROADCAST CAP_NET_RAW
CAP_SYSLOG CAP_BPF
AXOCLOUDCONNECTORS_IMAGE=ghcr.io/axoflow/axocloudconnectors:latest
AXOCLOUDCONNECTORS_PODMAN_ARGS=--log-level=warn
STORAGE_DIRECTORY=/var/lib/axocloudconnectors-elastic
AXOROUTER_ENDPOINT=127.0.0.1:4317
http_proxy=
https_proxy=
no_proxy=
AXOCLOUDCONNECTOR_DEVICE_ID=8f6c3653
AXOROUTER_TLS_INSECURE=true
ELASTICSEARCH_ENDPOINT=<YOUR_ELASTICSEARCH_ENDPOINT>
ELASTICSEARCH_API_KEY=<YOUR_ELASTICSEARCH_API_KEY>
ELASTICSEARCH_INDEX=<YOUR_INDEX_NAME>

# ELASTICSEARCH_START_AT=beginning
```

Replace:

- `<YOUR_ELASTICSEARCH_ENDPOINT>` with the URL of your Elasticsearch instance (for Elastic Cloud, it looks something like `https://my-elasticsearch-project-3784re.es.us-east-1.aws.elastic.cloud:443` for Elastic Cloud)
- `<YOUR_ELASTICSEARCH_API_KEY>` with your actual API key
- `<YOUR_INDEX_NAME>` with the name of the Elasticsearch index from which you want to retrieve data

By default, Axoflow Cloud Connector collects only data that arrived into the index after Axoflow Cloud Connector started for the first time. If you want to fetch all data from the index, uncomment the `# ELASTICSEARCH_START_AT=beginning` line.

2. Create the `/etc/containers/systemd/axorouter-cloudconnectors-elastic.container` file with the following content:

```
[Unit]
Description=CloudConnectors Elastic Container
Wants=network-online.target
After=network.target network-online.target
StartLimitIntervalSec=0
StartLimitBurst=0

[Install]
WantedBy=multi-user.target default.target

[Container]
User=${AXOCLOUDCONNECTORS_UID}
Group=${AXOCLOUDCONNECTORS_GID}

# Passing envvars into the container
EnvironmentFile=/etc/axorouter/cloudconnectors-elastic.env

Environment=AXO_SYSTEMD_UNIT_NAME=%n

ContainerName=axorouter-cloudconnectors-elastic
AddCapability=${AXOCLOUDCONNECTORS_CAPS}
Image=${AXOCLOUDCONNECTORS_IMAGE}

Volume=${STORAGE_DIRECTORY}:${STORAGE_DIRECTORY}:z

PodmanArgs=${AXOCLOUDCONNECTORS_PODMAN_ARGS}

Exec=
LogDriver=journald
Network=host
SecurityLabelDisable=true

[Service]
# for `podman run` (e.g. AXOCLOUDCONNECTORS_IMAGE)
EnvironmentFile=/etc/axorouter/cloudconnectors-elastic.env
RestartSec=2
Restart=always
```

3. Run the following commands to load and start the service and create the required data directory (for persistent position pointer and data buffer):

```
sudo mkdir -p /var/lib/axocloudconnectors-elastic
sudo systemctl daemon-reload
sudo systemctl start axorouter-cloudconnectors-elastic
```

4. Check the logs of the newly created service to verify it's running and there are no errors:

```
sudo journalctl -f -u axorouter-cloudconnectors-elastic
```

   The Axoflow Cloud Connector starts forwarding logs to the AxoRouter instance.

5. Add the appliance to AxoConsole.
   1. Open the AxoConsole and select **Topology**.
   2. Select **Add Item > Source**.
   3. If the Axoflow Cloud Connector is already sending logs to an AxoRouter instance that is registered in the AxoConsole, select **Detected**, then select the source.
   4. Enter the **IP address** and the **FQDN** of the Axoflow Cloud Connector instance.
   5. Select **Add**.
6. Create a Flow to route the data from the AxoRouter instance to a destination. You can use the Labels of this source to select messages from this source.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`elastic`
`product`	`meta.product`	`elasticsearch`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

7.12.15 Everpure

7.12.15.1 FlashArray

Send Everpure FlashArray logs to Axoflow, and look up the labels that Axoflow adds to its data.

FlashArray: All-flash array whose Purity OS sends alerts, REST logins, CLI audit records, and capacity data over syslog.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	value
`vendor`	`meta.vendor`	`everpure`
`product`	`meta.product`	`flasharray`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`purestorage:array:audit`	`infraops`
`purestorage:array`	`infraops`

The audit messages of the Purity OS (alerts, logins, and CLI commands) use the `purestorage:array:audit` sourcetype, the capacity reports (`space_utility_json.py`) use `purestorage:array` .

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `PURE_STORAGE` .

7.12.16 F5 Networks

Send logs from F5 Networks BIG-IP application delivery appliances to Axoflow.

7.12.16.1 BIG-IP

Send F5 Networks BIG-IP logs to Axoflow, and look up the labels that Axoflow adds to its data.

BIG-IP: Provides load balancing, traffic management, and application security for optimized service delivery.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Note that the device can be configured to send logs formatted as plain-text syslog, JSON, or key-value pairs. AxoRouter can automatically parse all flavors.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`f5`
`product`	`meta.product`	`bigip`
`service`	`meta.service.name`	`ASM` , `apmd` , `audit_forwarder` , `CROND` , `F5` , `httpd` , `mcpd` , `sshd` , `sshd(pam_audit)` , `systemd-journal` , `tmm` , `tmm1` , `tmm2` , `tmsh`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
f5:bigip:syslog	`netops`
f5:bigip:ltm:access_json	`netops`
f5:bigip:asm:syslog	`netops`
f5:bigip:apm:syslog	`netops`
f5:bigip:ltm:ssl:error	`netops`
f5:bigip:ltm:tcl:error	`netops`
f5:bigip:ltm:traffic	`netops`
f5:bigip:ltm:log:error	`netops`
f5:bigip:gtm:dns:request:irule	`netops`
f5:bigip:gtm:dns:response:irule	`netops`
f5:bigip:ltm:http:irule	`netops`
f5:bigip:ltm:failed:irule	`netops`
nix:syslog	`netops`

Tested with: Splunk Add-on for F5 BIG-IP

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `F5_BIGIP_APM` .

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.17 Forcepoint

Send logs from Forcepoint Next-Generation Firewall, Email Security, and WebProtect to Axoflow.

7.12.17.1 Email Security

Send Forcepoint Email Security logs to Axoflow, and look up the labels that Axoflow adds to its data.

Email Security: Protects email systems from spam, phishing, malware, and data exfiltration using advanced threat defense.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`forcepoint`
`product`	`meta.product`	`email`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype, source, and index settings:

sourcetype	index
`forcepoint:email:cef`	`email`

`forcepoint:email:kv`	`email`
`forcepoint:email:leef`	`email`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `FORCEPOINT_EMAILSECURITY`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

Earlier name/vendor

- Websense Email Security

7.12.17.2 Next-Generation Firewall (NGFW)

Send Forcepoint NGFW logs to Axoflow, and look up the labels that Axoflow adds to its data.

Next-Generation Firewall (NGFW): Next-gen firewall with deep packet inspection, policy enforcement, and integrated intrusion prevention.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`forcepoint`
`product`	`meta.product`	`firewall`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype, source, and index settings:

sourcetype	index
`websense:cg:cef`	`netproxy`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `FORCEPOINT_FIREWALL`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

Earlier name/vendor

- Websense Webprotect
- Forcepoint Webprotect
- Forcepoint Secure Web Gateway

7.12.17.3 WebProtect

Send Forcepoint Web Security logs to Axoflow, and look up the labels that Axoflow adds to its data.

WebProtect: Provides web traffic filtering, malware protection, and data loss prevention for secure internet access.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`forcepoint`
`product`	`meta.product`	`webprotect`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype, source, and index settings:

sourcetype	index
`websense:cg:cef`	`netproxy`
`websense:cg:kv`	`netproxy`
`websense:cg:leef`	`netproxy`

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

Earlier name/vendor

- Websense Firewall

7.12.18 Fortinet

Send logs from Fortinet FortiGate firewalls, FortiMail, FortiProxy, and FortiWeb to Axoflow.

7.12.18.1 FortiGate firewalls

Configure FortiGate next-generation firewalls to send logs to Axoflow, and look up the labels that Axoflow adds.

FortiGate firewalls: Enterprise firewall platform offering threat protection, VPN, and traffic filtering for secure networking.

The following sections show you how to configure FortiGate Next-Generation Firewall (NGFW) to send their log data to Axoflow.

> **CAUTION:**
>
> Make sure to set data forwarding on your appliances/servers as described in this guide. Different settings like alternate message formats or ports might be valid, but can result in data loss or incorrect parsing.

Prerequisites

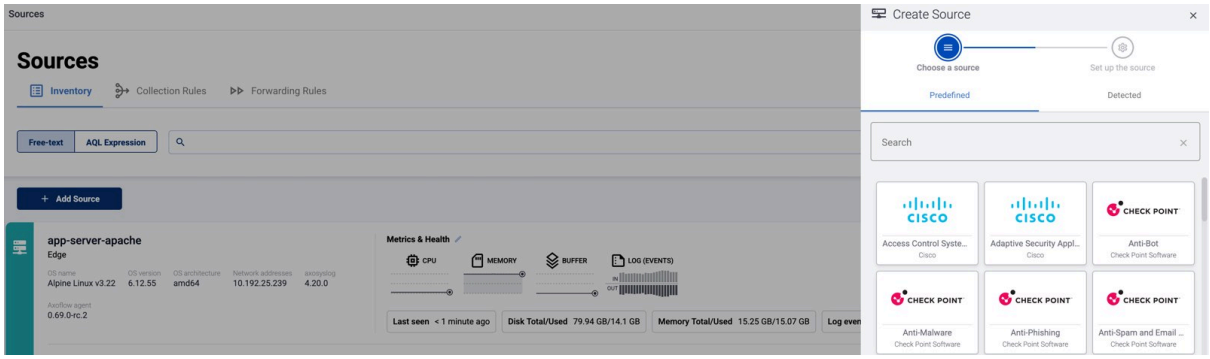
- You have administrative access to the firewall.
- The date, time, and time zone are correctly set on the firewall.
- You have an AxoRouter deployed and configured with a Syslog connector that has parsing and classification enabled (by default, every AxoRouter has such connectors). This device is going to receive the data from the firewall.
- You know the IP address the AxoRouter. To find it:

  1. Open the AxoConsole.
  2. Select the **Routers** or the **Topology** page.
  3. Select on AxoRouter instance that is going to receive the logs.
  4. Check the **Networks > Address** field.

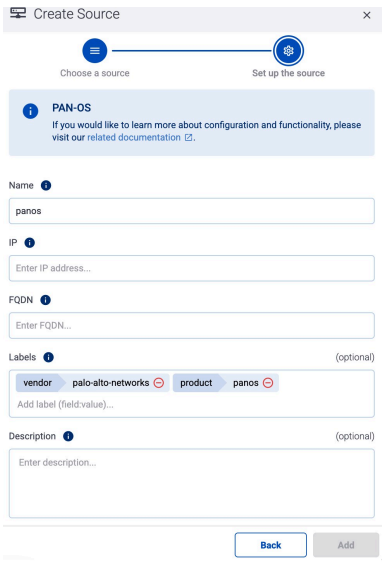
Steps

Note: The steps involving the FortiGate user interface are just for your convenience, for details, see the official FortiGate documentation.

1. Log in to your FortiGate device. You need administrator privileges to perform the configuration.
2. Register the address of your AxoRouter as an Address Object.
    1. Select **Log & Report > Log Settings > Global Settings**.
    2. Configure the following settings:
        - **Event Logging**: Click **All**.
        - **Local traffic logging**: Click **All**.
        - **Syslog logging**: Enable this option.
        - **IP address/FQDN**: Enter the address of your AxoRouter: `%axorouter-ip%`
    3. Click **Apply**.
3. Add the source to AxoConsole.
    1. Open the AxoConsole and select **Topology**.
    2. Select **Add Item > Source**.



    - If the source is actively sending data to an AxoRouter instance, select **Detected**, then select your source(s) and select **Add Selected Source(s)**.
    - Otherwise, select the vendor and product corresponding to your source from the **Predefined** sources, then enter the parameters of the source, like **IP address** and **FQDN**.



> **Note**
>
> **During log tapping, you can add hosts that are actively sending data to an AxoRouter instance by clicking Register source.**

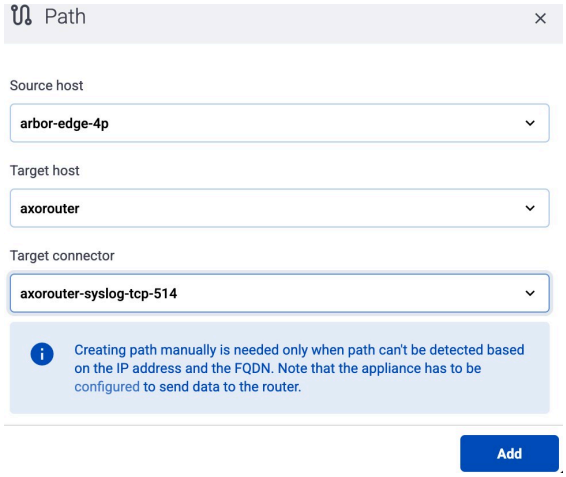
3. (Optional) Add custom labels as needed.

> **CAUTION:**
>
> AxoConsole can automatically raise an alert if a source that used to send data regularly stops sending data. To enable this alert for a source, add the `traffic_alerts:true` label to the source.

4. Select **Add**.

5. (Optional) **Add Path** manually. That's needed only when AxoConsole can't detect the path based on the IP address and the FQDN, and you haven't yet configured the source to send data to the router. If you've added the source from the **Detected** list, you can skip this step, as the path is created automatically.

> **Create a path between the source and the AxoRouter instance.**
>
> If you've added the source from the **Detected** list, you can skip this step, as the path is created automatically.
>
> 1. Select **Topology > Add Item > Path**.



2. If not set automatically, select the beginning of the path in the **Source host** field.
3. Select the target router or aggregator this source is sending its data to in the **Target host** field, for example, `axorouter`.
4. Select the **Target connector**. The connector determines how the destination receives the data (for example, using which protocol or port).
5. Select **Add**. The new path appears on the **Topology** page.



Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`fortinet`
`product`	`meta.product`	`fortigate`
`service`	`meta.service.name`	`fortigate`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`fortigate_event`	`netops`
`fortigate_traffic`	`netfw`
`fortigate_utm`	`netfw`

Tested with: Fortinet FortiGate Add-On for Splunk technical add-on

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `FORTINET_FIREWALL` .

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog` .

7.12.18.2 FortiMail

Send Fortinet FortiMail logs to Axoflow, and look up the labels that Axoflow adds to its data.

FortiMail: Secures inbound and outbound email with spam filtering, malware protection, and advanced threat detection.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`fortinet`
`product`	`meta.product`	`fortimail`
`service`	`meta.service.name`	`fortiweb`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`fml:log`	`email`

Tested with: FortiMail Add-on for Splunk technical add-on

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `FORTINET_FORTIMAIL`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.18.3 FortiProxy

Configure FortiProxy to send logs to AxoRouter over syslog, and look up the labels that Axoflow adds.

FortiProxy: Protects from internet threats like malware, phishing, and data loss by inspecting web traffic, filtering content, controlling applications, and enforcing security policies

To send logs from FortiProxy to AxoRouter, configure remote logging to a syslog server on FortiProxy. Use the IP address of AxoRouter as the syslog server IP address. Set FortiProxy to send logs in the `default` format. (If you need to send the logs in CEF format, contact our support team). For details, see Generic tips.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`fortinet`
`product`	`meta.product`	`fortiproxy`
`service`	`meta.service.name`	`fortigate`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`fortiproxy_anomaly`	`netfw`

`fortiproxy_event`	`netops`
`fortiproxy_traffic`	`netfw`
`fortiproxy_utm`	`netfw`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `FORTINET_WEBPROXY` .

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog` .

7.12.18.4 FortiWeb

Send Fortinet FortiWeb logs to Axoflow, and look up the labels that Axoflow adds to its data.

FortiWeb: Web application firewall protecting websites from attacks like XSS, SQL injection, and bot threats.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`fortinet`
`product`	`meta.product`	`fortiweb`
`service`	`meta.service.name`	`fortiweb`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`fwb_log`	`netops`
`fwb_attack`	`netids`
`fwb_event`	`netops`

`fwb_traffic` `netfw`

Tested with: Fortinet FortiWeb Add-On for Splunk technical add-on

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `FORTINET_FORTIWEB`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.19 Fortra

Forward IBM i security events from the Fortra Powertech SIEM Agent to Axoflow.

7.12.19.1 Powertech SIEM Agent for IBM i

Send Fortra Powertech SIEM Agent for IBM i logs to Axoflow, and look up the labels that Axoflow adds to its data.

Powertech SIEM Agent for IBM i: Monitors IBM i system activity and forwards security events for centralized analysis in SIEM platforms.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Note that the device can be configured to send logs formatted as CEF or LEEF. AxoRouter can automatically parse all flavors.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`forta`
`product`	`meta.product`	`powertech-siem-agent`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype, source, and index settings:

sourcetype	source	index

`PowerTech:SIEMAgent:cef`	`PowerTech:SIEMAgent`	`netops`
`PowerTech:SIEMAgent:leef`	`PowerTech:SIEMAgent`	`netops`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `FORTRA_POWERTECH_SIEM_AGENT` .

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog` .

Earlier name/vendor

Powertech Interact

7.12.20 General Unix/Linux host

Collect logs from common, non-vendor-specific services running on generic Unix and Linux hosts.

7.12.20.1 Generic Linux services

Send Generic UNIX/Linux host Generic Placeholder logs to Axoflow, and look up the labels that Axoflow adds to its data.

Generic Linux services: A generic placeholder for program classifications

These classifications include non-vendor specific services and applications commonly found on Linux/Unix hosts.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`nix`
`product`	`meta.product`	`generic`
`service`	`meta.service.name`	`bind`, `chronyd`, `cron`, `cupsd`, `dbus-daemon`, `dhcpd`, `dnsmasq`, `dnf`, `dockerd`, `NetworkManager`, `nginx`, `nxlog`, `rsyslogd`, `sshd`, `su`, `sudo`, `syslog-ng`, or `systemd`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

source	sourcetype	index
program:chron	nix:syslog	netops
program:chronyd	nix:syslog	netops
program:cupsd	nix:syslog	netops
program:dbus-daemon	nix:syslog	netops
program:dhcpd	isc:dhcpd	netipam
program:dnf	nix:syslog	netops
program:dockerd	nix:syslog	netops
program:dnsmasq	nix:syslog	netdns
program:named	isc:bind:network	netdns
program:NetworkManager	nix:syslog	netops
program:nxlog	nix:syslog	netops
program:rsyslogd	nix:syslog	netops
program:sshd	nix:syslog	netops
program:su	nix:syslog	netauth
program:sudo	nix:syslog	netauth
program:syslog-ng	nix:syslog	netops
program:systemd	nix:syslog	netops

Tested with: Splunk Add-on for Infoblox

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `BIND_DNS, ISC_DHCP, NIX_SYSTEM, or OPENSSH`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.21 Imperva

Send logs from Imperva Incapsula and SecureSphere to Axoflow.

7.12.21.1 Incapsula

Send Imperva Incapsula logs to Axoflow, and look up the labels that Axoflow adds to its data.

Incapsula: Cloud-based WAF, DDoS protection, and bot mitigation service for securing web applications and APIs.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`imperva`
`product`	`meta.product`	`incapsula`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype, source, and index settings:

sourcetype	source	index
`cef`	`Imperva:Incapsula`	`netwaf`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `IMPERVA_CEF`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.21.2 SecureSphere

Send Imperva SecureSphere logs to Axoflow, and look up the labels that Axoflow adds to its data.

SecureSphere: Provides on-prem web application, database, and file security with granular activity monitoring.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`imperva`
`product`	`meta.product`	`securesphere`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype, source, and index settings:

sourcetype	index
`imperva:waf:firewall:cef`	`netwaf`
`imperva:waf:security:cef`	`netwaf`
`imperva:waf`	`netwaf`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `IMPERVA_SECURESPHERE`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.22 Infoblox

Send DNS, DHCP, and IPAM logs from Infoblox NIOS to Axoflow.

7.12.22.1 NIOS

Send Infoblox NIOS logs to Axoflow, and look up the labels that Axoflow adds to its data.

NIOS: Delivers secure DNS, DHCP, and IPAM (DDI) services with centralized network control and automation.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`infloblox`
`product`	`meta.product`	`nios`
`service`	`meta.service.name`	`named` , `threat-protect-log`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index	source

`infoblox:threatprotect`	`netids`	`Infoblox:NIOS`
`infoblox:dns`	`netids`	`Infoblox:NIOS`

Tested with: Splunk Add-on for Infoblox

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `INFOBLOX, INFOBLOX_DNS` .

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog` .

7.12.23 Ivanti

Send logs from Ivanti Connect Secure VPN appliances to Axoflow.

7.12.23.1 Connect secure

Send Ivanti Connect Secure logs to Axoflow, and look up the labels that Axoflow adds to its data.

Connect secure: Provides dynamic IP address assignment and network configuration for DHCP-enabled devices.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`ivanti`
`product`	`meta.product`	`connect-secure`
`service`	`meta.service.name`	`PulseSecure`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype, source, and index settings:

sourcetype	source	index

`pulse:connectsecure`	`netfw`
`pulse:connectsecure:web`	`netproxy`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `IVANTI_CONNECT_SECURE`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

Earlier name/vendor

Pulse Connect Secure

7.12.24 Juniper

Send logs from Juniper routers, switches, and firewalls running Junos OS to Axoflow.

7.12.24.1 Junos OS

Send Juniper Networks Junos OS logs to Axoflow, and look up the labels that Axoflow adds to its data.

Junos OS: Junos OS is the network operating system for Juniper physical and virtual networking and security products.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`juniper`
`product`	`meta.product`	`junos`
`service`	`meta.service.name`	`eswd` , `ifinfo` , `mcsnoopd` , `mgd` , `mib2d` , `rpd` , `RT_AAMW` , `RT_FLOW` , `RT_IDP` , `RT_SECINTEL` , `RT_UTM` , `tfeb0`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`juniper:junos:aamw:structured`	`netfw`
`juniper:junos:firewall`	`netfw`
`juniper:junos:firewall`	`netids`
`juniper:junos:firewall:structured`	`netfw`
`juniper:junos:firewall:structured`	`netids`
`juniper:junos:idp`	`netids`
`juniper:junos:idp:structured`	`netids`
`juniper:legacy`	`netops`
`juniper:junos:secintel:structured`	`netfw`
`juniper:junos:snmp`	`netops`
`juniper:structured`	`netops`

Tested with: Splunk Add-on for Juniper

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `JUNIPER_JUNOS`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.25 Kaspersky

Send logs from Kaspersky Endpoint Security to Axoflow for processing and routing.

7.12.25.1 Endpoint Security

Send Kaspersky Endpoint Security logs to Axoflow, and look up the labels that Axoflow adds to its data.

Endpoint Security: Protects endpoints from malware, ransomware, and intrusions with antivirus, firewall, and threat detection.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Note that the device can be configured to send logs formatted as plain-text syslog, CEF, or LEEF. AxoRouter can automatically parse all flavors.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`kaspersky`
`product`	`meta.product`	`endpoint_security`
`service`	`meta.service.name`	`KES|11.0.0.0` , `KL_Threat_Feed_Service_v2`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Note that the device can be configured to send plain syslog text, LEEF, or CEF-formatted output.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`kaspersky:cef`	`epav`
`kaspersky:es`	`epav`
`kaspersky:gnrl`	`epav`
`kaspersky:klau`	`epav`
`kaspersky:klbl`	`epav`
`kaspersky:klmo`	`epav`
`kaspersky:klna`	`epav`
`kaspersky:klpr`	`epav`
`kaspersky:klsr`	`epav`
`kaspersky:leef`	`epav`
`kaspersky:sysl`	`epav`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `KASPERSKY_ENDPOINT`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.26 Kubernetes

Collect logs from Kubernetes clusters, including NGINX Ingress, with Telemetry Controller and send them to Axoflow.

7.12.26.1 NGINX Ingress

Send NGINX Ingress Controller logs from Kubernetes to the OpenTelemetry connector of AxoRouter, and look up the labels Axoflow adds.

NGINX Ingress: Ingress NGINX Controller

Configure your log collector (for example, Telemetry Controller) to send data to an OpenTelemetry Connector of an AxoRouter instance using the OTLP/gRPC protocol.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`kubernetes`
`product`	`meta.product`	`nginx`
`service`	`meta.service.name`	`ingress-nginx`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

7.12.26.2 Telemetry Controller

Send Kubernetes logs collected by Telemetry Controller to the OpenTelemetry connector of AxoRouter, and look up the labels Axoflow adds.

Telemetry Controller: Telemetry Controller

Configure Telemetry Controller to send data to an OpenTelemetry Connector of an AxoRouter instance.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`kubernetes`
`product`	`meta.product`	`telemetry-controller`
`kubernetes_namespace`	`meta.kubernetes.namespace`	dynamic
`kubernetes_container`	`meta.kubernetes.container`	dynamic

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

7.12.27 Microsoft

Collect data from Windows hosts, Azure Event Hubs, and Microsoft Cloud App Security (MCAS) with Axoflow.

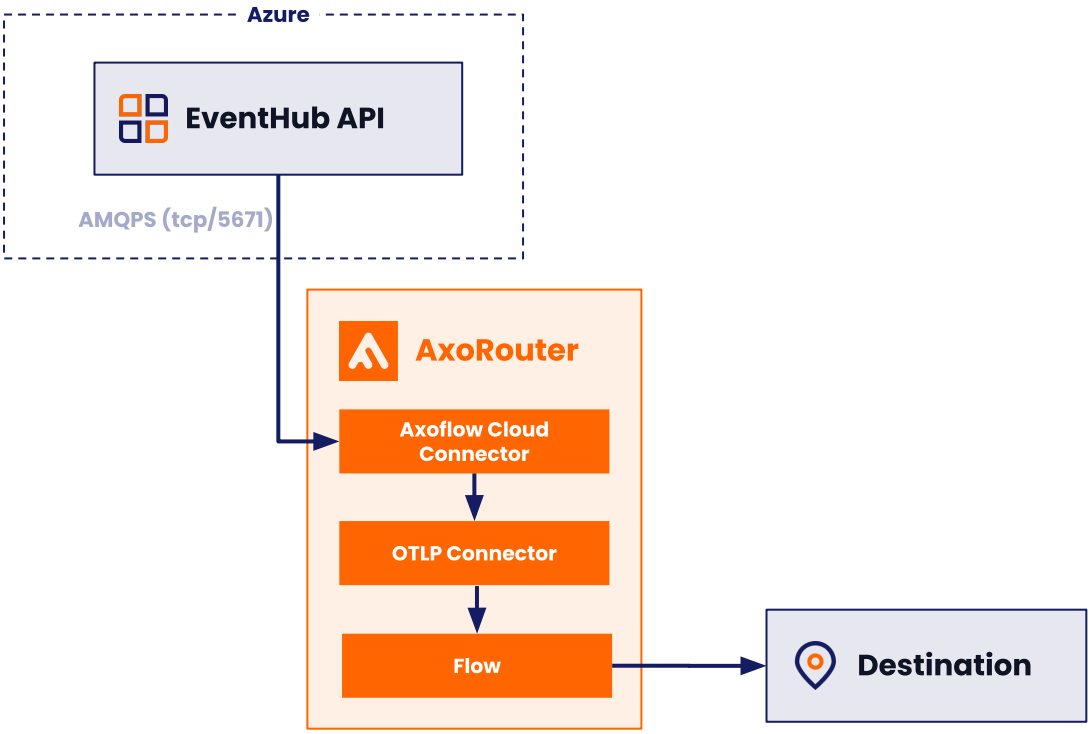
7.12.27.1 Azure Event Hubs

Collect data from Azure Event Hubs with a Cloud Connector, and look up the labels and field mappings for each log type.

Azure Event Hubs: Big data streaming platform to ingest and process events.

Axoflow can collect data from your Azure Event Hubs deployment with an Axoflow Cloud Connector. At a high level, the process looks like this:

- Create a **Connector Rule** on the AxoRouter that deploys an Axoflow Cloud Connector and collects the data from your Event Hub. Axoflow Cloud Connector is a simple container that runs on an AxoRouter instance.
- Configure a Flow on AxoConsole that processes and routes the collected data to your destination (for example, Splunk or another SIEM).



Prerequisites

- An AxoRouter instance that:
  - Was provisioned with **Enable Cloudconnectors** turned on.

- Can reach Azure Event Hubs on TCP port 5671. Adjust your firewall rules and the access rules of the data source as needed.
- You know the IP address the AxoRouter. To find it:

  1. Open the AxoConsole.
  2. Select the **Routers** or the **Topology** page.
  3. Select on AxoRouter instance that is going to receive the logs.
  4. Check the **Networks > Address** field.

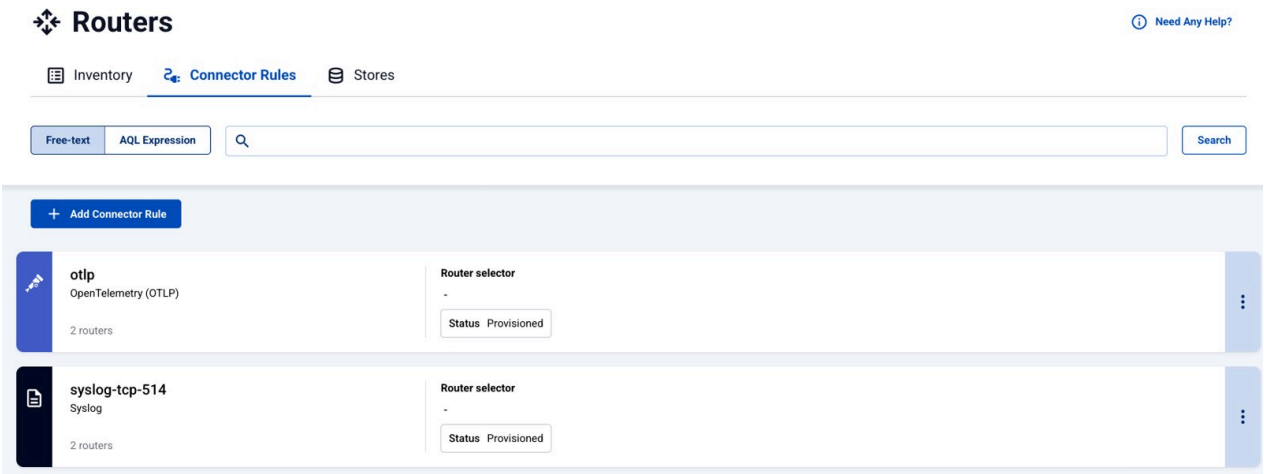
- An Azure account with an active subscription.
- An Event Hubs connection string that includes a Shared Access Key (SAS token) and an EntityPath name.

> **Note**
>
> **To collect data from multiple EntityPaths, you must create a separate connector rule in AxoConsole for each EntityPath.**

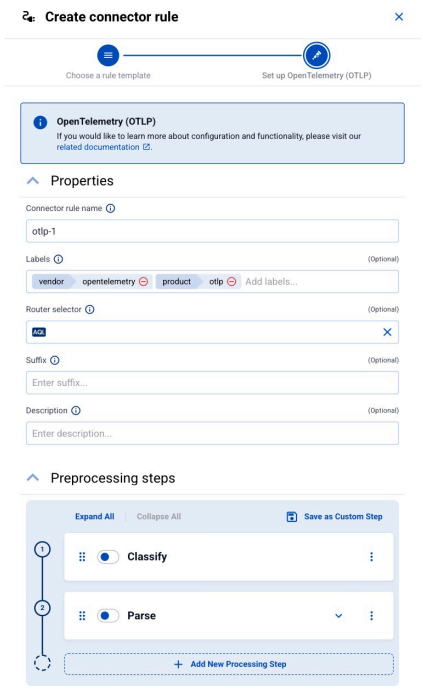
Configure Axoflow Cloud Connector on AxoRouter

To collect data from Azure Event Hubs, complete the following steps.

1. Select **Routers > Connector Rules > Add Connector Rule**. (Alternatively, you can select **Add Connector > Add a connector rule** on the **Connectors** page of an AxoRouter host.)



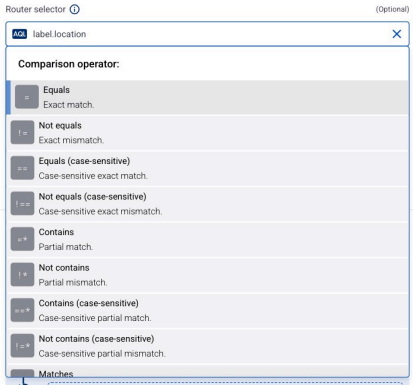
2. Select **Azure Event Hubs**.
3. Enter a name for the connector rule into the **Connector rule name** field.



4. (Optional) Add labels to the connector rule. You will be able to use these labels in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps. For details about the message schema and the related connector-specific fields, see the `meta.connector` object.

5. Set the **Router selector** for the connector rule. The selector determines which AxoRouter instances get a connector from this connector rule.

   > **CAUTION:**
   >
   > Each connector rule for Azure Event Hubs must match exactly one AxoRouter. Use the `name` or `guid` AQL selector to make sure it matches only one AxoRouter. Otherwise several Axoflow Cloud Connector instances collect the same data, and Axoflow receives every message multiple times.



   You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

   - If you leave the router selector empty, it matches every AxoRouter instance.

- To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.
- If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

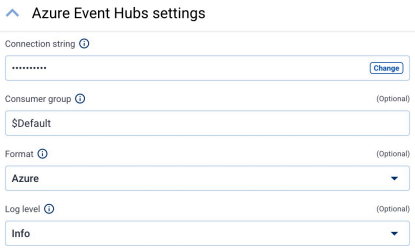
6. (Optional) Enter a **Suffix** for the connector rule. AxoConsole uses the suffix in the name of the connector instances that it creates on the AxoRouter hosts. The name of the connector instance follows the `<name-of-AxoRouter>-<suffix>` schema. For example, if the name of a matching AxoRouter instance is `axorouter-1`, and the suffix of the rule is `custom`, the name of the connector on that AxoRouter is `axorouter-1-custom`.

   If the **Suffix** field is empty, AxoConsole uses the name of the connector rule instead. For example, if the name of the connector rule is `otlp-1`, the name of the connector is `axorouter-1-otlp-1`.

7. (Optional) Enter a description for the rule.

8. Enter your Azure Event Hubs connection string into the **Connection string** field.

   The connection string must include a Shared Access Key (SAS token) and an EntityPath name.



9. (Optional) If needed, set the **Consumer group** of the data. By default, it's set to `$Default`.

10. (Optional) If you fetch non-standard or custom data from Event Hubs, change **Format** to **Raw**.

11. Select **Add**. The Axoflow Cloud Connector starts forwarding logs to the AxoRouter instance.

12. Verify that the Axoflow Cloud Connector is running.

    1. Select **Routers**, then select the AxoRouter you added the Axoflow Cloud Connector to.
    2. Select **Services** and check that **axoflow-cloudconnector.service** is running without errors.

13. Create a Flow to route the data from the AxoRouter instance to a destination. You can use the Labels of this source to select its messages.
14. To fetch data from another EntityPath, repeat this procedure from step 1. Make sure that only one AxoRouter fetches data from each EntityPath.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`microsoft`
`product`	`meta.product`	`azure-event-hubs`
`azure_event_hubs_category`	`log.body.category`	The log category of the event being logged.

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Field mapping

The Axoflow Cloud Connector maps the fields of the Azure common log format to OpenTelemetry attributes. The `azure.*` attributes arrive as dotted keys in the message body, for example, `azure.correlation.id` becomes `log.body["correlation.id"]`.

The following table shows which of these fields Axoflow uses when it processes and normalizes each log category:

Azure field	Message field	Audit	Graph activity	Provisioning	Signin
`category`	`log.body.category`	✓	✓	✓	✓
`operationName`	`log.body["operation.name"]`	✓	✓		✓
`operationVersion`	`log.body["operation.version"]`	✓	✓		✓
`correlationId`	`log.body["correlation.id"]`	✓	✓		✓
`tenantId`	`log.body["tenant.id"]`	✓	✓		✓

`durationMs`	`log.body.duration`	✓	✓	✓
`resultSignature`	`log.body["result.signature"]`	✓	✓	✓
`properties`	`log.body.properties`	✓	✓	✓
`identity`	`log.body.identity`	✓		✓
`resultType`	`log.body["result.type"]`	✓		✓
`location`	`log.body["cloud.region"]`		✓	✓
`callerIpAddress`	`log.body["network.peer.address"]`		✓	✓
`resultDescription`	`log.body["result.description"]`			✓

> **Note**
>
> - Axoflow always sets the `cloud.provider` field to `azure`.
> - The `properties` field carries most of the data, for example, the signin logs read about 40 subfields from it, such as `deviceDetail`, `status.errorCode`, and `location.geoCoordinates`.

Event Hubs Audit logs labels

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`microsoft`
`product`	`meta.product`	`azure-event-hubs`
`service`	`meta.service.name`	`audit`

Event Hubs Graph Activity logs labels

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`microsoft`
`product`	`meta.product`	`azure-event-hubs`
`service`	`meta.service.name`	`graphactivity`

Event Hubs Provisioning logs labels

Analytics label	Message field	Value

`vendor`	`meta.vendor`	`microsoft`
`product`	`meta.product`	`azure-event-hubs`
`service`	`meta.service.name`	`provisioning`

Event Hubs Signin logs labels

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`microsoft`
`product`	`meta.product`	`azure-event-hubs`
`service`	`meta.service.name`	`signin`

Sending data to Splunk

When you send the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`mscs:azure:eventhub`	`azure-activity`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `AZURE_EVENTHUB` .

7.12.27.2 Cloud App Security (MCAS)

Send Microsoft Cloud App Security (MCAS) logs to Axoflow, and look up the labels that Axoflow adds to its data.

Cloud App Security (MCAS): Monitors cloud app usage, detects anomalies, and enforces security policies across SaaS services.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`microsoft`
`product`	`meta.product`	`cas`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	source	index
`cef`	`microsoft:cas`	`main`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `MICROSOFT_DEFENDER_CLOUD_ALERTS`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.27.3 Windows hosts

Choose between the agent-based and agentless ways to collect Windows event logs, and look up the labels that Axoflow adds.

Windows hosts: Event logs from core services like security, system, DNS, and DHCP for operational and forensic analysis.

To collect event logs from Microsoft Windows hosts, Axoflow supports both agent-based and agentless methods.

- For a collector agent, we recommend using the Axoflow OpenTelemetry Collector distribution. For details, see Windows host - agent based solution.
- To use an agentless solution, see Windows Event Collector (WEC).

Labels

Labels assigned to data received from Windows hosts depend on how AxoRouter receives the data. For details, see Windows host - agent based solution and Windows Event Collector (WEC).

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`windows:eventlog:snare`	`oswin`
`windows:eventlog:xml`	`oswin`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `WINEVTLOG, WINEVTLOG_XML, WINDOWS_DHCP, WINDOWS_DNS` .

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following tables.

- Event logs from the Security channel: `SecurityEvent`
- Event logs from the System channel: `SecurityEvent`
- Other event logs: `WindowsEvent`

7.12.28 MikroTik

Send logs from MikroTik routers running RouterOS to Axoflow.

7.12.28.1 RouterOS

Send MikroTik RouterOS logs to Axoflow, and look up the labels that Axoflow adds to its data.

RouterOS: Router operating system providing firewall, bandwidth management, routing, and hotspot functionality.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`mikrotik`
`product`	`meta.product`	`routeros`
`service`	`meta.service.name`	`forward`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index

`routeros`	`netfw`
`routeros`	`netops`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `MIKROTIK_ROUTER`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.29 NetFlow Logic

Send flow data processed by NetFlow Logic NetFlow Optimizer to Axoflow.

7.12.29.1 NetFlow Optimizer

Configure NetFlow Optimizer to send its data to Axoflow, and look up the labels that Axoflow adds.

NetFlow Optimizer: Aggregates and transforms flow data (NetFlow, IPFIX) into actionable security and performance insights.

The following sections show you how to configure NetFlow Optimizer to send their log data to Axoflow.

> **CAUTION:**
>
> Make sure to set data forwarding on your appliances/servers as described in this guide. Different settings like alternate message formats or ports might be valid, but can result in data loss or incorrect parsing.

Prerequisites

- You have administrative access to NetFlow Optimizer.
- You have an AxoRouter deployed and configured with a Syslog connector that has parsing and classification enabled (by default, every AxoRouter has such connectors). This device is going to receive the data from NetFlow Optimizer.
- You know the IP address the AxoRouter. To find it:

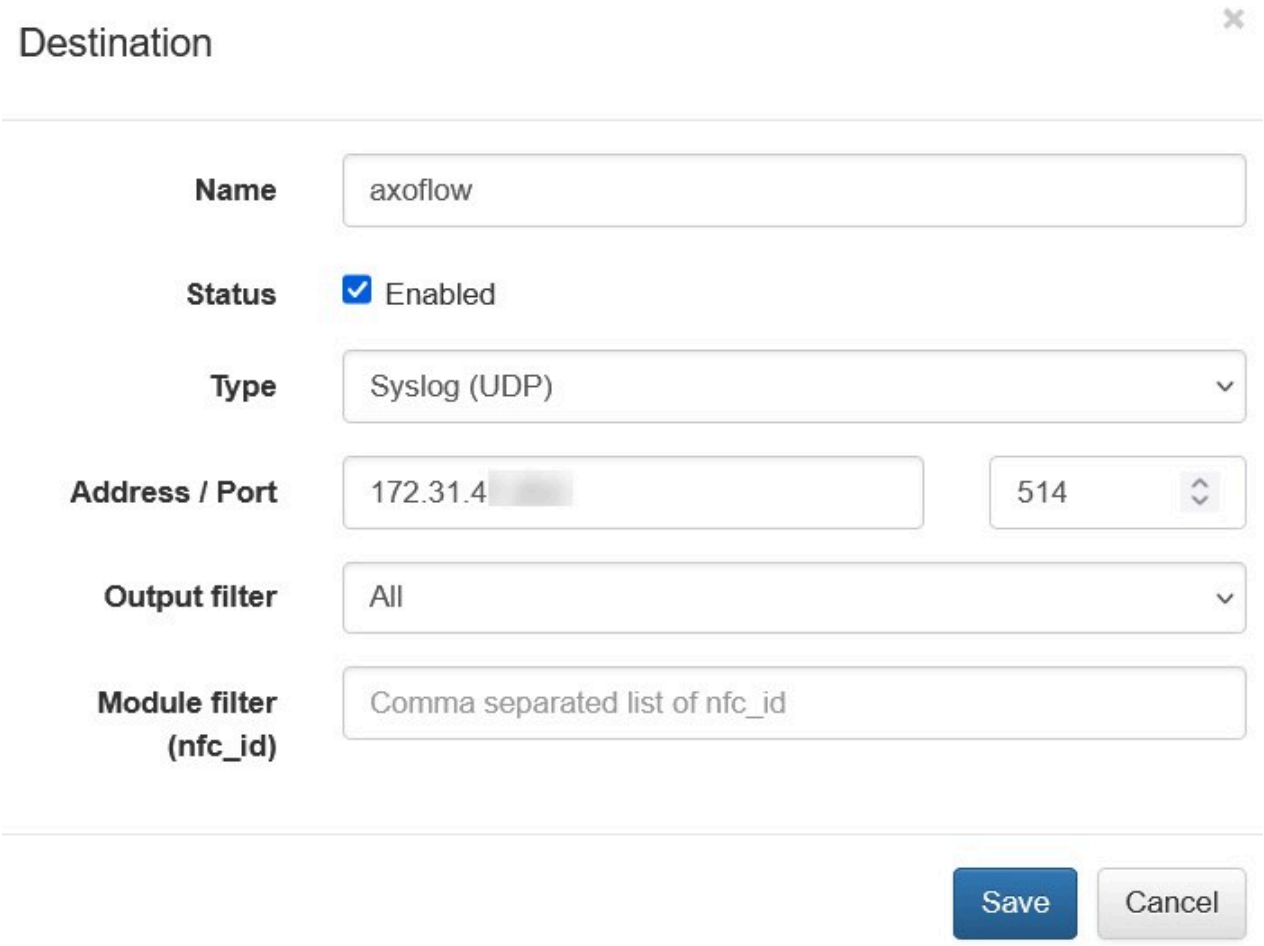
  1. Open the AxoConsole.
  2. Select the **Routers** or the **Topology** page.
  3. Select on AxoRouter instance that is going to receive the logs.
  4. Check the **Networks > Address** field.

Steps

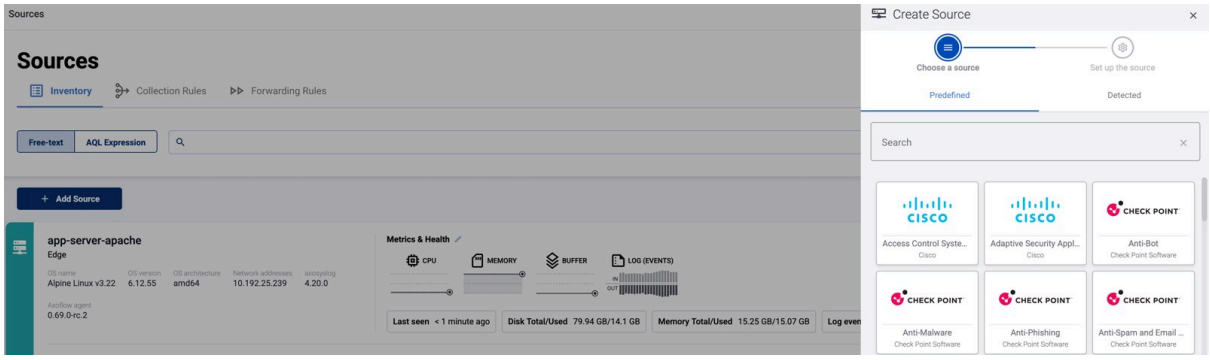
Note: The steps involving the NetFlow Optimizer user interface are just for your convenience, for details, see the official documentation.

1. Log in to NetFlow Optimizer.
2. Select **Outputs**, then click the plus sign to add an output to NetFlow Optimizer.

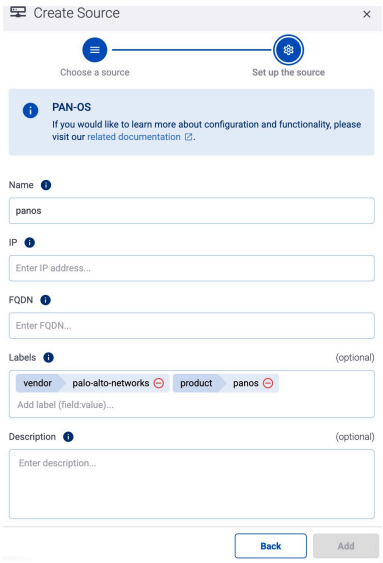
3. Configure a **Syslog (UDP)** output:
   - **Name**: Enter a name for the output, for example, `Axoflow`.
   - **Address**: The IP address of the AxoRouter instance where you want to send the messages.
   - **Port**: Set this parameter to 514.



4. Click **Save**.
5. Add the source to AxoConsole.
   1. Open the AxoConsole and select **Topology**.
   2. Select **Add Item > Source**.



- If the source is actively sending data to an AxoRouter instance, select **Detected**, then select your source(s) and select **Add Selected Source(s)**.
- Otherwise, select the vendor and product corresponding to your source from the **Predefined** sources, then enter the parameters of the source, like **IP address** and **FQDN**.



> ## Note
>
> **During log tapping, you can add hosts that are actively sending data to an AxoRouter instance by clicking Register source.**

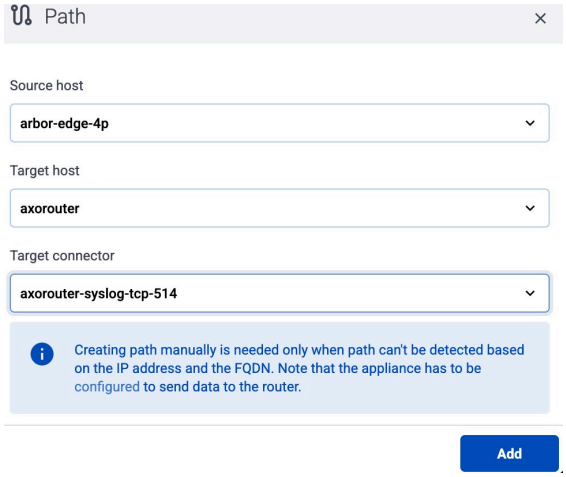
3. (Optional) Add custom labels as needed.

> **CAUTION:**
>
> AxoConsole can automatically raise an alert if a source that used to send data regularly stops sending data. To enable this alert for a source, add the `traffic_alerts:true` label to the source.

4. Select **Add**.

5. (Optional) **Add Path** manually. That's needed only when AxoConsole can't detect the path based on the IP address and the FQDN, and you haven't yet configured the source to send data to the router. If you've added the source from the **Detected** list, you can skip this step, as the path is created automatically.

> **Create a path between the source and the AxoRouter instance.**
>
> If you've added the source from the **Detected** list, you can skip this step, as the path is created automatically.
>
> 1. Select **Topology > Add Item > Path**.



2. If not set automatically, select the beginning of the path in the **Source host** field.
3. Select the target router or aggregator this source is sending its data to in the **Target host** field, for example, `axorouter`.
4. Select the **Target connector**. The connector determines how the destination receives the data (for example, using which protocol or port).
5. Select **Add**. The new path appears on the **Topology** page.



Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value

`vendor`	`meta.vendor`	`netflow`
`product`	`meta.product`	`optimizer`
`service`	`meta.service.name`	Network Conversations Monitor, SNMP Custom OID Sets Monitor, Sampling Monitor, SNMP Information Monitor, DNS Service Monitor, DNS Users Monitor, Network Subnets Monitor, Asset Access Monitor, Services Performance Monitor, Top Bandwidth Consumers for Cisco ASA, Top Traffic Destinations for Cisco ASA, Top Policy Violators for Cisco ASA, Top Hosts with most Connections for Cisco ASA, Outbound Mail Spammers Monitor, Inbound Mail Spammers Monitor, Unauthorized Mail Servers Monitor, Rejected Emails Monitor, Top Bandwidth Consumers for Palo Alto Networks Firewall, Top Traffic Destinations for Palo Alto Networks Firewall, Hosts with Most Policy Violations for Palo Alto Networks Firewall, Most Active Hosts for Palo Alto Networks Firewall, Bandwidth Consumption per Application for Palo Alto Networks Firewall, Bandwidth Consumption per Application/User for Palo Alto Networks, Top Applications Traffic Monitor, Top Applications Host Pairs Monitor, Visitors by Country, Botnet C&C Traffic Monitor, Custom Threat lists Monitor, Host Reputation Monitor, Threat Feeds Traffic Monitor, TCP Health Monitor, Network Conversations Monitor, Top Connections Monitor, Top Pairs Monitor, CBQoS Monitor, Traffic by Autonomous Systems, Top Traffic Monitor, Top Packets Monitor, SNMP Custom OID Sets Monitor, Top Bandwidth Consumers for NSX Distributed Firewall, Top Traffic Destinations for NSX Distributed Firewall, Top Policy Violators for NSX Distributed Firewall, Top Hosts with most Connections for NSX Distributed Firewall, Top Host VM:Host Pairs, Top VM:Host Traffic Monitor, AWS VPC Flow logs, Micro-segmentation Top Pairs Monitor, AWS Top Traffic Monitor, GCP VPC Flow Logs, GCP Top Traffic Monitor, Azure NSG Flow Logs, Cisco AVC Top Applications Monitor, Cisco AVC Bandwidth Consumption Monitor, Azure Top Traffic Monitor, Cisco AnyConnect Top Traffic Monitor, SNMP Traps Monitor, Auto-discovery Reporter, Top Traffic Monitor Geo City, Top Traffic Monitor Geo Country, Flow Data

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`flowintegrator`	`flowintegrator`

7.12.30 Netgate

Send firewall logs from Netgate pfSense appliances to Axoflow.

7.12.30.1 pfSense

Send Netgate pfSense logs to Axoflow, and look up the labels that Axoflow adds to its data.

pfSense: Open-source firewall and router platform with VPN, traffic shaping, and intrusion detection capabilities.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`netgate`
`product`	`meta.product`	`pfsense`
`service`	`meta.service.name`	`filterlog`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index

`pfsense:filterlog`	`netfw`
`pfsense:<program>`	`netops`

The `pfsense:<program>` variant is simply a generic linux event that is generated by the underlying OS on the appliance.

Tested with: TA-pfsense

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `PFSENSE`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.31 Netmotion

Send logs from Netmotion to Axoflow and see which labels Axoflow adds to them.

7.12.31.1 Netmotion

Send Netmotion logs to Axoflow, and look up the labels that Axoflow adds to its data.

Netmotion: Provides secure, optimized remote access with performance monitoring for mobile and distributed workforces.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`netmotion`
`product`	`meta.product`	`netmotion`
`service`	`meta.service.name`	`LocalityServer` , `nmreporting.exe`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index

`netmotion:reporting`	`netops`
`netmotion:mobilityserver:nm_mobilityanalyticsappdata`	`netops`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `NETMOTION` .

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog` .

7.12.32 NETSCOUT

Send logs from NETSCOUT Arbor Edge Defense (AED) and Arbor Pravail (APS) to Axoflow.

7.12.32.1 Arbor Edge Defense (AED)

Send NETSCOUT Arbor Edge Defense (AED) logs to Axoflow, and look up the labels that Axoflow adds to its data.

Arbor Edge Defense (AED): Edge-based DDoS protection and threat mitigation system to block attacks before they enter the network.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`netscout`
`product`	`meta.product`	`arbor-edge`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype, source, and index settings:

sourcetype	source	index
`netscout:aed`	`netscout:aed`	`netids`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `ARBOR_EDGE_DEFENSE`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.32.2 Arbor Pravail (APS)

Send NETSCOUT Pravail (APS) logs to Axoflow, and look up the labels that Axoflow adds to its data.

Arbor Pravail (APS): Monitors and mitigates advanced persistent threats and malware with inline packet inspection.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`netscout`
`product`	`meta.product`	`arbor-pravail`
`service`	`meta.service.name`	`arbor-networks-aps` , `pravail`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype, source, and index settings:

sourcetype	source	index
`netscout:aps`	`netscout:aps`	`netids`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the

name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `ARBOR_EDGE_DEFENSE`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

Earlier name/vendor

- Arbor Networks Pravail (APS)

7.12.33 Omnissa

Send logs from Omnissa Horizon (formerly VMware Horizon) to Axoflow.

Earlier name/vendor

VMware Horizon

7.12.33.1 Horizon

Send Omnissa Horizon logs to Axoflow, and look up the labels that Axoflow adds to its data.

Horizon: Horizon is a platform for managing and delivering virtual desktops and applications to end-users across a variety of devices and locations.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`omnissa`
`product`	`meta.product`	`horizon`
`service`	`meta.service.name`	`view`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`vmware:horizon`	`infraops`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the

name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `VMWARE_HORIZON`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

Earlier name/vendor

VMware Horizon

7.12.34 OpenText

Send logs from OpenText ArcSight and Self Service Password Reset (SSPR) to Axoflow.

7.12.34.1 ArcSight

Send OpenText ArcSight logs to Axoflow, and look up the labels that Axoflow adds to its data.

ArcSight: SIEM platform for collecting, correlating, and analyzing security event data across IT environments.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`opentext`
`product`	`meta.product`	`arcsight`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype, source, and index settings:

sourcetype	source	index
`cef`	`ArcSight:ArcSight`	`main`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `ARCSIGHT_CEF`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

Earlier name/vendor

MicroFocus ArcSight

7.12.34.2 Self Service Password Reset (SSPR)

Send OpenText Self Service Password Reset (SSPR) logs to Axoflow, and look up the labels that Axoflow adds to its data.

Self Service Password Reset (SSPR): Allows users to securely reset their own passwords without IT assistance, reducing helpdesk load.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`opentext`
`product`	`meta.product`	`sspr`
`service`	`meta.service.name`	`SSPR`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype, source, and index settings:

sourcetype	index
`sspr`	`netauth`

Earlier name/vendor

NetIQ Self Service Password Reset

7.12.35 Palo Alto Networks

Send logs from Palo Alto Networks firewalls running PAN-OS and from Cortex XSOAR to Axoflow.

7.12.35.1 Cortex XSOAR

Send Palo Alto Networks Cortex XSOAR logs to Axoflow, and look up the labels that Axoflow adds to its data.

Cortex XSOAR: Security orchestration, automation, and response platform for threat detection and incident management.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`palo-alto-networks`
`product`	`meta.product`	`cortex-xsoar`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype, source, and index settings:

sourcetype	source	index
`cef`	`tim:cef`	`infraops`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `PAN_XSOAR`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

Earlier name/vendor

Threat Intelligence Management (TIM)

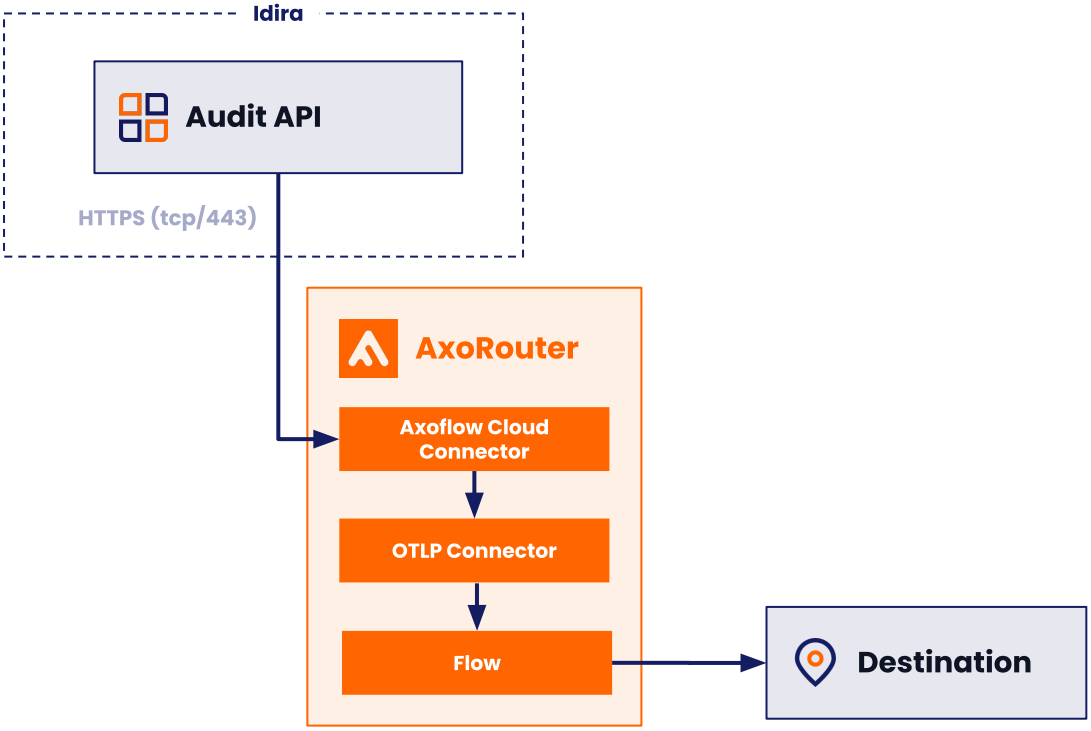
7.12.35.2 Idira

Collect data from Palo Alto Networks Idira with a Cloud Connector, and look up the labels Axoflow adds.

Idira: Polls the Idira audit API for identity security platform events, authenticating with OAuth2 client credentials.

Axoflow can collect data from your Idira deployment with an Axoflow Cloud Connector. At a high level, the process looks like this:

- Create a **Connector Rule** on the AxoRouter that deploys an Axoflow Cloud Connector and collects the data from Idira. Axoflow Cloud Connector is a simple container that runs on an AxoRouter instance.
- Configure a Flow on AxoConsole that processes and routes the collected data to your destination (for example, Splunk or another SIEM).



Prerequisites

- An AxoRouter instance that:
  - Was provisioned with **Enable Cloudconnectors** turned on.
  - Can reach the Idira audit API and the Identity Administration token endpoint on TCP port 443. Adjust your firewall rules and the access rules of the data source as needed.
  - You know the IP address the AxoRouter. To find it:

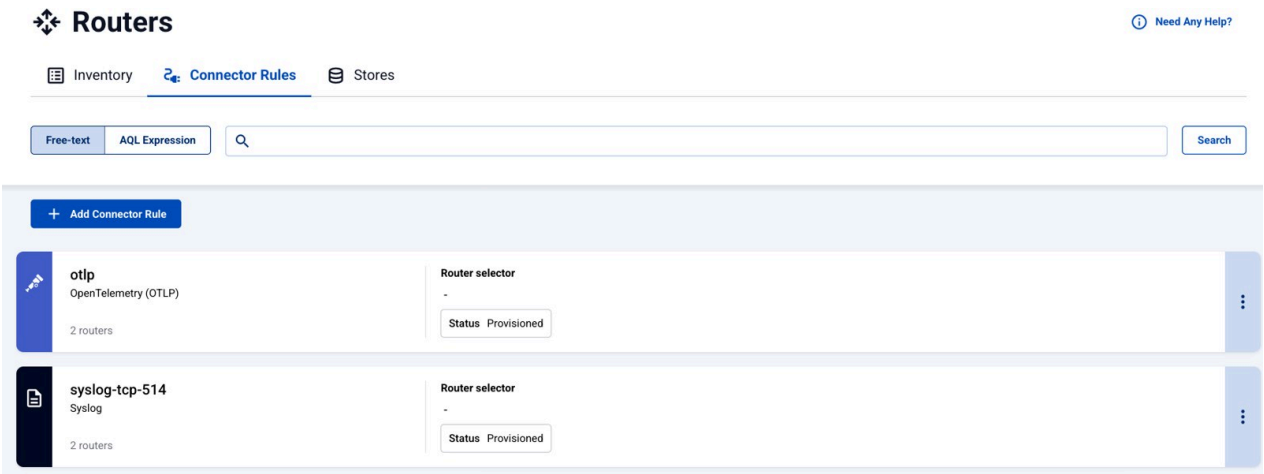
1. Open the AxoConsole.
2. Select the **Routers** or the **Topology** page.
3. Select on AxoRouter instance that is going to receive the logs.
4. Check the **Networks > Address** field.

- An Idira tenant with the SIEM integration enabled.
- The base URL of the Idira audit API, and a SIEM integration API key for it.
- An OAuth2 confidential client registered in Identity Administration, and the client ID and client secret of this client. The client must be allowed to request the `isp.audit.events:read` scope.

Configure Axoflow Cloud Connector on AxoRouter

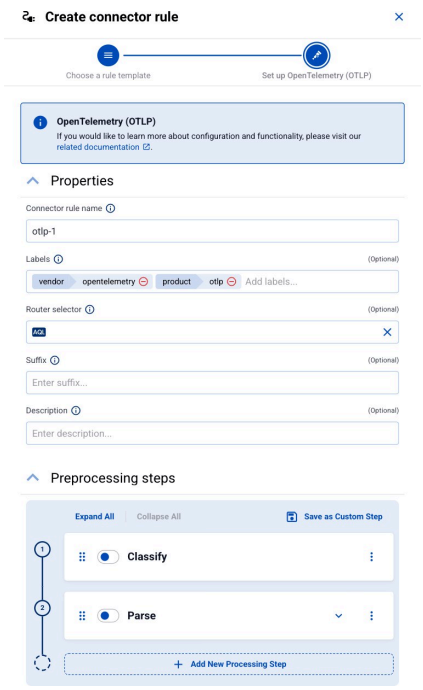
To collect data from Idira, complete the following steps.

1. Select **Routers > Connector Rules > Add Connector Rule**. (Alternatively, you can select **Add Connector > Add a connector rule** on the **Connectors** page of an AxoRouter host.)



2. Select **Idira**.

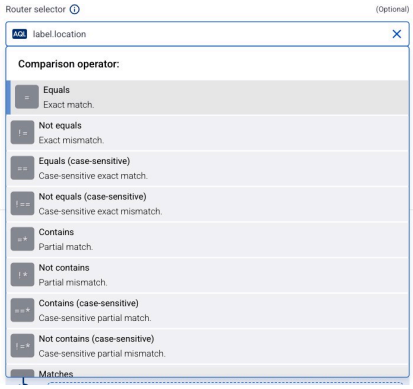
3. Enter a name for the connector rule into the **Connector rule name** field.



4. (Optional) Add labels to the connector rule. You will be able to use these labels in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps. For details about the message schema and the related connector-specific fields, see the `meta.connector` object.

5. Set the **Router selector** for the connector rule. The selector determines which AxoRouter instances get a connector from this connector rule.

   > **CAUTION:**
   >
   > Each connector rule for Idira must match exactly one AxoRouter. Use the `name` or `guid` AQL selector to make sure it matches only one AxoRouter. Otherwise several Axoflow Cloud Connector instances collect the same data, and Axoflow receives every message multiple times.



   You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

   - If you leave the router selector empty, it matches every AxoRouter instance.

- To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.
- If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

6. (Optional) Enter a **Suffix** for the connector rule. AxoConsole uses the suffix in the name of the connector instances that it creates on the AxoRouter hosts. The name of the connector instance follows the `<name-of-AxoRouter>-<suffix>` schema. For example, if the name of a matching AxoRouter instance is `axorouter-1`, and the suffix of the rule is `custom`, the name of the connector on that AxoRouter is `axorouter-1-custom`.

   If the **Suffix** field is empty, AxoConsole uses the name of the connector rule instead. For example, if the name of the connector rule is `otlp-1`, the name of the connector is `axorouter-1-otlp-1`.

7. (Optional) Enter a description for the rule.

8. Configure how Axoflow Cloud Connector reaches the Idira audit API.

   - **Endpoint**: The base URL of the Idira audit API.
   - **API key**: The SIEM integration API key. Axoflow Cloud Connector sends it in the `x-api-key` header of every request.
   - **Token URL**: The OAuth2 token endpoint of Identity Administration, in the `https://<identity_fqdn>/OAuth2/Token/<web_app_id>` format.
   - **Client ID** and **Client secret**: The credentials of the OAuth2 confidential client that Axoflow Cloud Connector uses to obtain the access token.

9. (Optional) Select **More options** to change the advanced options of the connector, for example, how often it polls the API.

10. Select **Add**. The Axoflow Cloud Connector starts polling the Idira audit API and forwarding the collected events to the AxoRouter instance.

11. Verify that the Axoflow Cloud Connector is running.

    1. Select **Routers**, then select the AxoRouter you added the Axoflow Cloud Connector to.
    2. Select **Services** and check that **axoflow-cloudconnector.service** is running without errors.

12. Create a Flow to route the data from the AxoRouter instance to a destination. You can use the Labels of this source to select its messages.

Advanced options

Select **More options** on the connector rule to set the following options.

- **Scopes**: The OAuth2 scopes that Axoflow Cloud Connector requests for the access token. **Default** requests the `isp.audit.events:read` scope. Select **Custom** to request a different set of scopes. An empty custom list requests a token with no scopes at all.
- **Poll interval**: How often Axoflow Cloud Connector creates a new query. The Idira API accepts one query per minute, so Axoflow rejects shorter intervals. Default value: `1m`
- **Initial lookback**: How far back the first poll queries when no checkpoint is stored yet. Set it to `0` to collect only the events that arrive after the connector starts. Default value: `5m`

  Axoflow Cloud Connector checkpoints the position of the poll cursor, so after a restart it resumes the collection where the last poll stopped. The **Initial lookback** option applies only when there is no stored checkpoint.
- **Page size**: The number of events that Axoflow Cloud Connector fetches per results page. Minimum value: `1`. Default value: `500`
- **Application codes**: Restricts the query to the listed application codes. Leave it empty to collect the events of every application.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`idira`
`product`	`meta.product`	`idira`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

7.12.35.3 Palo Alto firewalls

Configure Palo Alto Networks next-generation firewalls to send logs to Axoflow, and look up the labels that Axoflow adds.

Palo Alto firewalls: Firewall operating system delivering network security features including traffic control and threat prevention.

The following sections show you how to configure Palo Alto Networks Next-Generation Firewall devices to send their log data to Axoflow.

> **CAUTION:**
>
> Make sure to set data forwarding on your appliances/servers as described in this guide. Different settings like alternate message formats or ports might be valid, but can result in data loss or incorrect parsing.

Prerequisites

- You have administrative access to the firewall.
- The date, time, and time zone are correctly set on the firewall.
- You have an AxoRouter deployed and configured with a Syslog connector that has parsing and classification enabled (by default, every AxoRouter has such connectors). This device is going to receive the data from the firewall.
- You know the IP address the AxoRouter. To find it:

  1. Open the AxoConsole.
  2. Select the **Routers** or the **Topology** page.
  3. Select on AxoRouter instance that is going to receive the logs.
  4. Check the **Networks > Address** field.

Steps

Note: The steps involving the Palo Alto Networks Next-Generation Firewall user interface are just for your convenience, for details, see the official PAN-OS® documentation.

1. Log in to your firewall device. You need administrator privileges to perform the configuration.

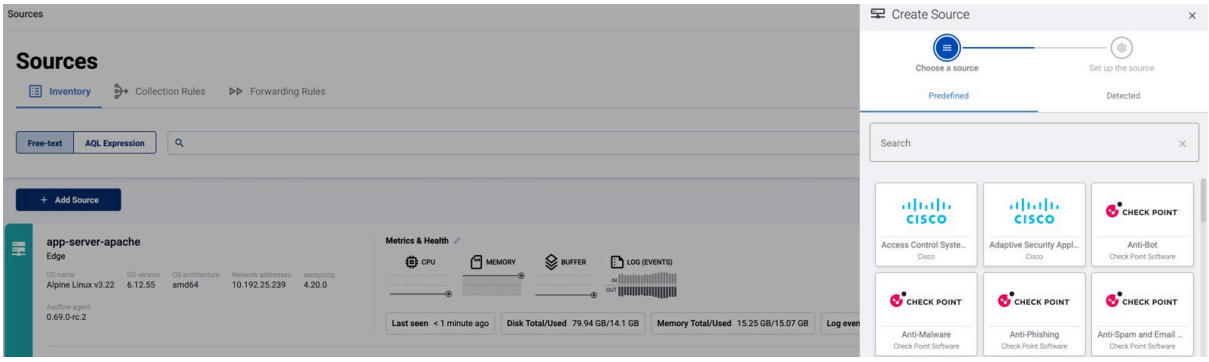
2. Configure a Syslog server profile.

   1. Select **Device > Server Profiles > Syslog**.

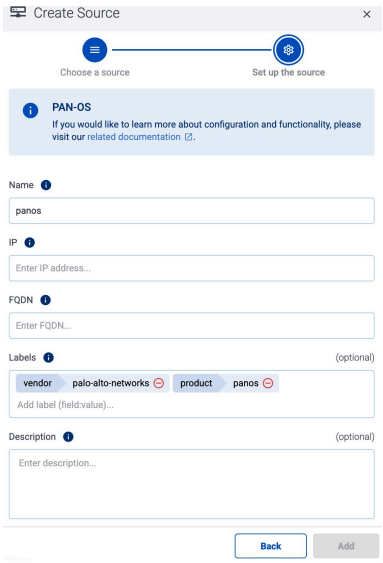
2. Click **Add** and enter a **Name** for the profile, for example, `axorouter`.
3. Configure the following settings:
   - **Syslog Server**: Enter the IP address of your AxoRouter: `%axorouter-ip%`
   - **Transport**: Select **TCP** or **TLS**.
   - **Port**: Set the port to `601`. (This is needed for the recommended IETF log format. If for some reason you need to use the BSD format, set the port to `514`.)
   - **Format**: Select **IETF**.
   - **Syslog logging**: Enable this option.
4. Click **OK**.

3. Configure syslog forwarding for Traffic, Threat, and WildFire Submission logs. For details, see Configure Log Forwarding the official PAN-OS® documentation.
   1. Select **Objects > Log Forwarding**.
   2. Click **Add**.
   3. Enter a **Name** for the profile, for example, `axoflow`.
   4. For each log type, severity level, or WildFire verdict, select the **Syslog** server profile.
   5. Click **OK**.
   6. Assign the log forwarding profile to a security policy to trigger log generation and forwarding.
   7. Select **Policies > Security** and select a policy rule.
   8. Select **Actions**, then select the **Log Forwarding** profile you created (for example, `axoflow`).
   9. For Traffic logs, select one or both of the **Log at Session Start** and **Log At Session End** options.
   10. Click **OK**.
4. Configure syslog forwarding for System, Config, HIP Match, and Correlation logs.
   1. Select **Device > Log Settings**.
   2. For System and Correlation logs, select each Severity level, select the **Syslog** server profile, and click **OK**.
   3. For Config, HIP Match, and Correlation logs, edit the section, select the **Syslog** server profile, and click **OK**.
5. Click **Commit**.
6. Add the source to AxoConsole.
   1. Open the AxoConsole and select **Topology**.

2. Select **Add Item > Source**.



- If the source is actively sending data to an AxoRouter instance, select **Detected**, then select your source(s) and select **Add Selected Source(s)**.
- Otherwise, select the vendor and product corresponding to your source from the **Predefined** sources, then enter the parameters of the source, like **IP address** and **FQDN**.



> **Note**
>
> **During log tapping, you can add hosts that are actively sending data to an AxoRouter instance by clicking Register source.**

3. (Optional) Add custom labels as needed.

> **CAUTION:**
>
> AxoConsole can automatically raise an alert if a source that used to send data regularly stops sending data. To enable this alert for a source, add the `traffic_alerts:true` label to the source.

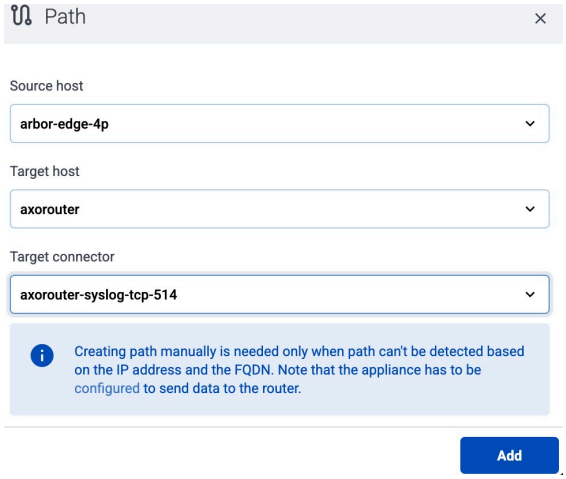
4. Select **Add**.

5. (Optional) **Add Path** manually. That's needed only when AxoConsole can't detect the path based on the IP address and the FQDN, and you haven't yet configured the source to send data to the router. If you've added the source from the **Detected** list, you can skip this step, as the path is created automatically.

   Create a path between the source and the AxoRouter instance.

   If you've added the source from the **Detected** list, you can skip this step, as the path is created automatically.

   1. Select **Topology > Add Item > Path**.

   2. If not set automatically, select the beginning of the path in the **Source host** field.
   3. Select the target router or aggregator this source is sending its data to in the **Target host** field, for example, `axorouter`.
   4. Select the **Target connector**. The connector determines how the destination receives the data (for example, using which protocol or port).
   5. Select **Add**. The new path appears on the **Topology** page.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`palo-alto-networks`
`product`	`meta.product`	`panos`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`pan:audit`	`netops`
`pan:globalprotect`	`netfw`
`pan:hipmatch`	`epintel`
`pan:traffic`	`netfw`
`pan:threat`	`netproxy`
`pan:system`	`netops`

Tested with: Palo Alto Networks Add-on for Splunk technical add-on

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `PAN_FIREWALL` .

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `CommonSecurityLog`.

7.12.36 Ping Identity

Collect logs from Ping Identity PingAccess and send them to Axoflow.

7.12.36.1 PingAccess

Send Ping Identity PingAccess logs to Axoflow, and look up the labels that Axoflow adds to its data.

PingAccess: A centralized access security solution which provides secure access to applications and APIs down to the URL level

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`ping-identity`
`product`	`meta.product`	`pingaccess`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	source	index
`source.engine`	`pingaccess`	`netauth`

Tested with: SecureAuth IdP Splunk App

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `PING_ACCESS`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.37 Progress

Send logs from the Progress Flowmon Anomaly Detection System (ADS) to Axoflow.

7.12.37.1 Flowmon Anomaly Detection System (ADS)

Send Progress Flowmon ADS logs to Axoflow, and look up the labels that Axoflow adds to its data.

Flowmon Anomaly Detection System (ADS): Detects network anomalies and threats through flow-based behavior analysis and machine learning.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`progress`
`product`	`meta.product`	`flowmon-ads`
`service`	`meta.service.name`	`ADS`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index

`flowmon-ads` `netids`

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

Earlier name/vendor

- Flowmon Networks

7.12.38 Riverbed

Send logs from Riverbed SteelHead and SteelConnect appliances to Axoflow.

7.12.38.1 SteelConnect

Send Riverbed SteelConnect logs to Axoflow, and look up the labels that Axoflow adds to its data.

SteelConnect: Software-defined WAN solution for centralized network management and secure cloud connectivity.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`riverbed`
`product`	`meta.product`	`steelconnect`
`service`	`meta.service.name`	`sport` , `scm_*`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index

`riverbed:steelconnect`	`netops`
`riverbed:syslog` (`sport` messages)	`netops`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `RIVERBED` .

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog` .

7.12.38.2 SteelHead

Send Riverbed SteelHead logs to Axoflow, and look up the labels that Axoflow adds to its data.

SteelHead: WAN optimization appliance that accelerates application performance and reduces bandwidth usage.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`riverbed`
`product`	`meta.product`	`steelhead`
`service`	`meta.service.name`	`/opt/hal/bin/hal` , `alarmd` , `crld` , `domaind` , `hald_model` , `link_control` , `lumberjack_rbt-upgrader` , `lumberjack_rbt` , `mdreq` , `mgmtd` , `ocd` , `periodic_raidcheck` , `pm` , `rcud` , `restd` , `rgpd` , `rscored-upgrader` , `rscored` , `rstild-upgrader` , `rstild` , `sched` , `statsd` , `wdt` , `webasd`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`riverbed:steelhead`	`netops`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `STEELHEAD`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.39 RSA

Send authentication logs from RSA Authentication Manager to Axoflow.

7.12.39.1 Authentication Manager

Send RSA Authentication Manager logs to Axoflow, and look up the labels that Axoflow adds to its data.

Authentication Manager: Manages two-factor authentication using RSA SecurID tokens for secure access to enterprise resources.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`rsa`
`product`	`meta.product`	`authentication-manager`
`service`	`meta.service.name`	`SecureAuth2`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index

`rsa:securid:admin:syslog`	`netauth`
`rsa:securid:system:syslog`	`netauth`
`rsa:securid:runtime:syslog`	`netauth`
`rsa:securid:syslog`	`netauth`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `RSA_AUTH_MANAGER` .

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog` .

7.12.40 rsyslog

Send logs from rsyslog to AxoRouter over the syslog protocol. Axoflow handles rsyslog as a generic syslog source.

Axoflow treats rsyslog sources as a generic syslog source. To send data from rsyslog to Axoflow, just configure rsyslog to send data to an AxoRouter instance using the syslog protocol.

Note that even if rsyslog is acting as a relay (receiving data from other clients and forwarding them to AxoRouter), on the **Topology** page it will be displayed as a data source.

Prerequisites

- You have administrative access to the device running rsyslog.
- The date, time, and time zone are correctly set on the appliance.
- You have an AxoRouter deployed and configured with a Syslog connector that has parsing and classification enabled (by default, every AxoRouter has such connectors). This device is going to receive the data from the source.
- You know the IP address the AxoRouter. To find it:

  1. Open the AxoConsole.
  2. Select the **Routers** or the **Topology** page.
  3. Select on AxoRouter instance that is going to receive the logs.
  4. Check the **Networks > Address** field.

7.12.41 SecureAuth

Collect logs from the SecureAuth Identity Platform and send them to Axoflow.

7.12.41.1 Identity Platform

Send SecureAuth Identity Platform logs to Axoflow, and look up the labels that Axoflow adds to its data.

Identity Platform: Delivers identity and access management with adaptive authentication and single sign-on capabilities.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`secureauth`
`product`	`meta.product`	`idp`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`secureauth:idp`	`netops`

Tested with: SecureAuth IdP Splunk App

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `SECUREAUTH_SSO` .

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog` .

7.12.42 Skyhigh Security

Send logs from the Skyhigh Security Secure Web Gateway to Axoflow.

7.12.42.1 Secure Web Gateway

Send Skyhigh Security Secure Web Gateway logs to Axoflow, and look up the labels that Axoflow adds to its data.

Secure Web Gateway: Inspects and filters web traffic to protect against malware, enforce policies, and prevent data loss.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`skyhigh`
`product`	`meta.product`	`secure-web-gateway`
`service`	`meta.service.name`	`mwg`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	source	index

`mcafee:wg:leef`	`mcafee:wg`	`netproxy`
`mcafee:wg:kv`	`mcafee:wg`	`netproxy`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `MCAFEE_WEBPROXY` .

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog` .

Earlier name/vendor

McAfee Secure Web Gateway

7.12.43 SonicWall

Send firewall and VPN logs from SonicWall appliances to Axoflow.

7.12.43.1 SonicWall

Configure SonicWall firewalls running SonicOS 6.x or 7.x to send logs to Axoflow, and look up the labels that Axoflow adds.

SonicWall: Delivers firewall, VPN, and deep packet inspection to protect networks from cyber threats and intrusions.

The following sections show you how to configure SonicWall firewalls to send their log data to Axoflow.

> **CAUTION:**
>
> Make sure to set data forwarding on your appliances/servers as described in this guide. Different settings like alternate message formats or ports might be valid, but can result in data loss or incorrect parsing.

Prerequisites

- You have administrative access to the firewall.
- The date, time, and time zone are correctly set on the firewall.
- You have an AxoRouter deployed and configured with a Syslog connector that has parsing and classification enabled (by default, every AxoRouter has such connectors). This device is going to receive the data from the firewall.
- You know the IP address the AxoRouter. To find it:

  1. Open the AxoConsole.
  2. Select the **Routers** or the **Topology** page.
  3. Select on AxoRouter instance that is going to receive the logs.
  4. Check the **Networks > Address** field.

Steps for SonicOS 7.x

Note: The steps involving the SonicWall user interface are just for your convenience, for details, see the official SonicWall documentation.

1. Log in to your SonicWall device. You need administrator privileges to perform the configuration.

2. Register the address of your AxoRouter as an Address Object.
    1. Select **MENU > OBJECT**.
    2. Select **Match Objects > Addresses > Address objects**.
    3. Click **Add Address**.
    4. Configure the following settings:
        - **Name**: Enter a name for the AxoRouter, for example, `AxoRouter`.
        - **Zone Assignment**: Select the correct zone.
        - **Type**: Select **Host**.
        - **IP Address**: Enter the IP address of your AxoRouter: `%axorouter-ip%`
    5. Click **Save**.
3. Set your AxoRouter as a syslog server.
    1. Navigate to **Device > Log > Syslog**.
    2. Select the **Syslog Servers** tab.
    3. Click **Add**.
    4. Configure the following options:
        - **Name or IP Address**: Select the Address Object of AxoRouter.
        - **Server Type**: Select **Syslog Server**.
        - **Syslog Format**: Select **Enhanced**.

        If your Syslog server does not use default port 514, type the port number in the **Port** field.

        By default, AxoRouter accepts data on the following ports (unless you've modified the default connector rules):

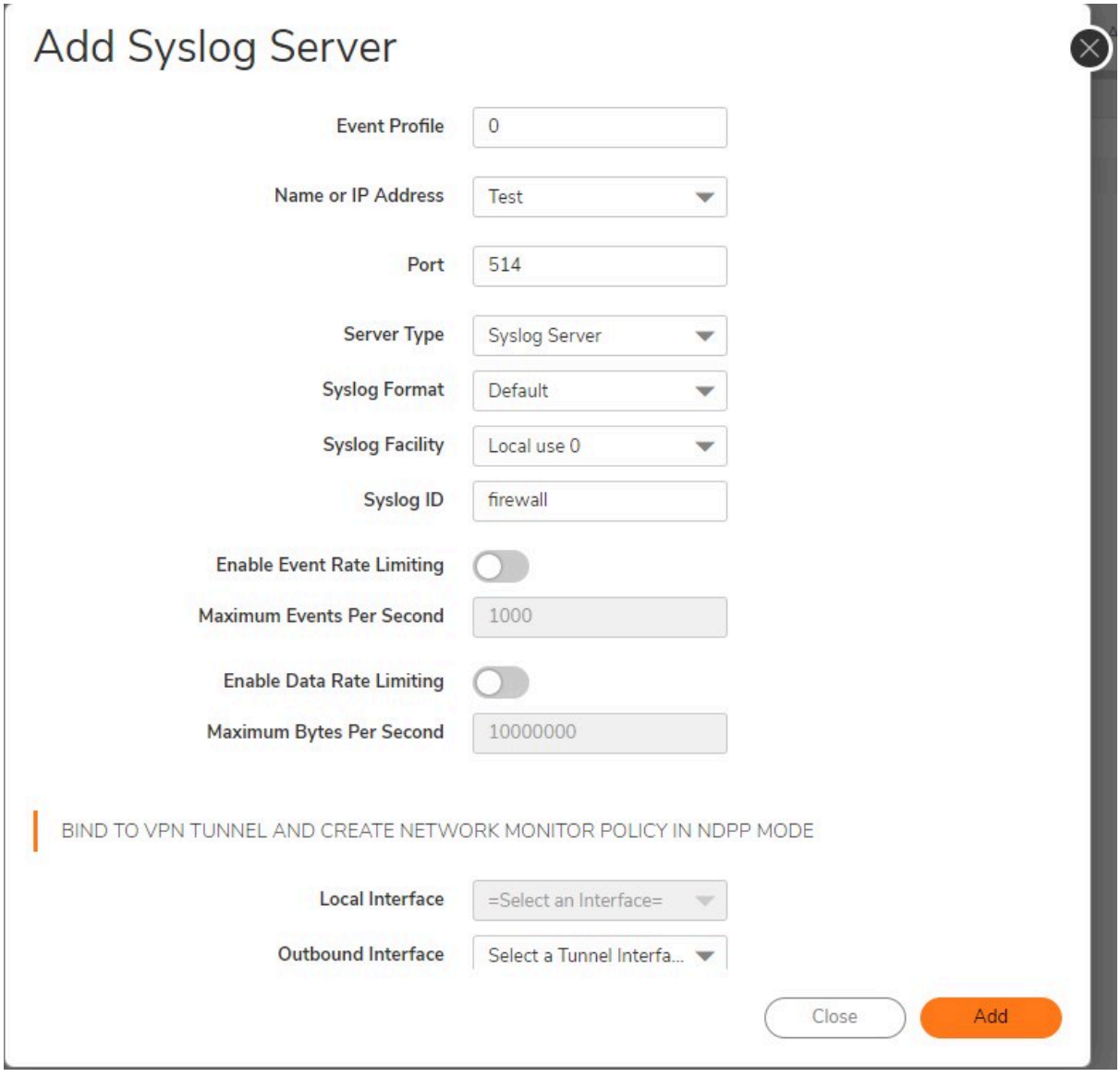
        - 514 UDP and TCP for RFC3164 (BSD-syslog) and RFC5424 (IETF-syslog) formatted traffic. AxoRouter automatically recognizes and handles both formats.
        - 601 TCP (RFC 3195 - Reliable Delivery for syslog) for RFC5424 (IETF-syslog) and RFC3164 (BSD-syslog) formatted traffic. AxoRouter automatically recognizes and handles both formats.
        - 6514 TCP for TLS-encrypted syslog traffic (RFC 5425).
        - 4317 TCP for OpenTelemetry log data.
        - 5085 TCP for the Windows Event Collector (WEC) service.
        - 8088 TCP for the Splunk HEC receiver (optional, only if you forward Splunk HEC traffic to AxoRouter).

- 9200 TCP for the Elasticsearch Bulk API connector (optional, only if you forward data from Elastic agents to AxoRouter).

To receive data on other ports or other protocols, configure other connector rules for the AxoRouter host.

For TLS-encrypted syslog connections, create a new connector rule or edit an existing one, and configure the keys and certificates needed to encrypt the connections. For details, see Syslog.

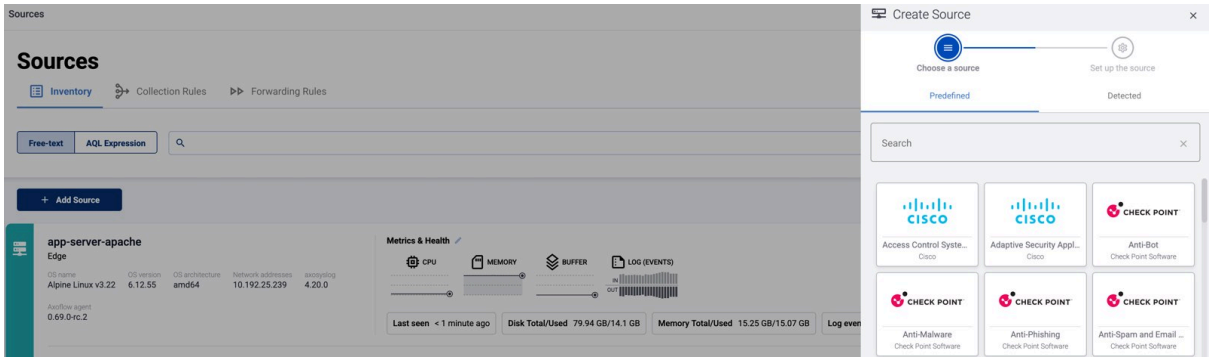
> **Note**
>
> **When you create a connector rule on a new port, open that port in the firewall of of every corresponding AxoRouter as well.**



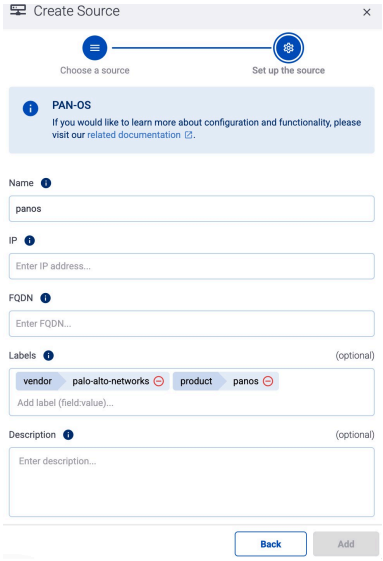
4. Add the source to AxoConsole.

    1. Open the AxoConsole and select **Topology**.

2. Select **Add Item > Source**.



- If the source is actively sending data to an AxoRouter instance, select **Detected**, then select your source(s) and select **Add Selected Source(s)**.
- Otherwise, select the vendor and product corresponding to your source from the **Predefined** sources, then enter the parameters of the source, like **IP address** and **FQDN**.



> **Note**
>
> **During log tapping, you can add hosts that are actively sending data to an AxoRouter instance by clicking Register source.**

3. (Optional) Add custom labels as needed.

> **CAUTION:**
>
> AxoConsole can automatically raise an alert if a source that used to send data regularly stops sending data. To enable this alert for a source, add the `traffic_alerts:true` label to the source.

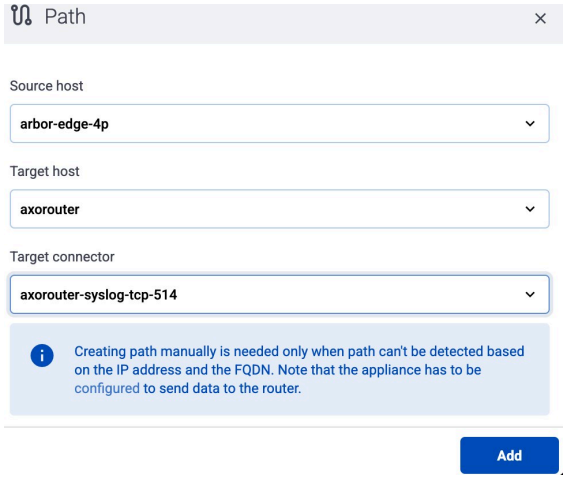
4. Select **Add**.

5. (Optional) **Add Path** manually. That's needed only when AxoConsole can't detect the path based on the IP address and the FQDN, and you haven't yet configured the source to send data to the router. If you've added the source from the **Detected** list, you can skip this step, as the path is created automatically.

   Create a path between the source and the AxoRouter instance.

   If you've added the source from the **Detected** list, you can skip this step, as the path is created automatically.

   1. Select **Topology > Add Item > Path**.

   2. If not set automatically, select the beginning of the path in the **Source host** field.
   3. Select the target router or aggregator this source is sending its data to in the **Target host** field, for example, `axorouter`.
   4. Select the **Target connector**. The connector determines how the destination receives the data (for example, using which protocol or port).
   5. Select **Add**. The new path appears on the **Topology** page.

Steps for SonicOS 6.x

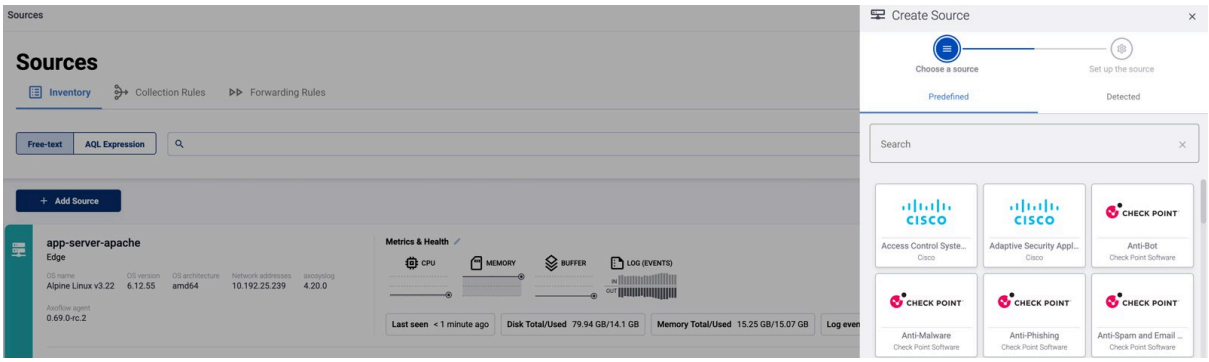
Note: The steps involving the SonicWall user interface are just for your convenience, for details, see the official SonicWall documentation.

1. Log in to your SonicWall device. You need administrator privileges to perform the configuration.
2. Register the address of your AxoRouter as an Address Object.

1. Select **MANAGE > Policies > Objects > Address Objects**.
2. Click **Add**.
3. Configure the following settings:
   - **Name**: Enter a name for the AxoRouter, for example, `AxoRouter`.
   - **Zone Assignment**: Select the correct zone.
   - **Type**: Select **Host**.
   - **IP Address**: Enter the IP address of your AxoRouter: `%axorouter-ip%`
4. Click **Add**.

3. Set your AxoRouter as a syslog server.
   1. Navigate to **MANAGE > Log Settings > SYSLOG**.
   2. Click **ADD**.
   3. Configure the following options:
      - **Syslog ID**: Enter an ID for the firewall. This ID will be used as the hostname in the log messages.
      - **Name or IP Address**: Select the Address Object of AxoRouter.
      - **Server Type**: Select **Syslog Server**.
      - Enable the **Enhanced Syslog Fields Settings**.
   4. Click **OK**.

4. Add the source to AxoConsole.
   1. Open the AxoConsole and select **Topology**.
   2. Select **Add Item > Source**.



   - If the source is actively sending data to an AxoRouter instance, select **Detected**, then select your source(s) and select **Add Selected Source(s)**.

- Otherwise, select the vendor and product corresponding to your source from the **Predefined** sources, then enter the parameters of the source, like **IP address** and **FQDN**.



> ## Note
>
> **During log tapping, you can add hosts that are actively sending data to an AxoRouter instance by clicking Register source.**

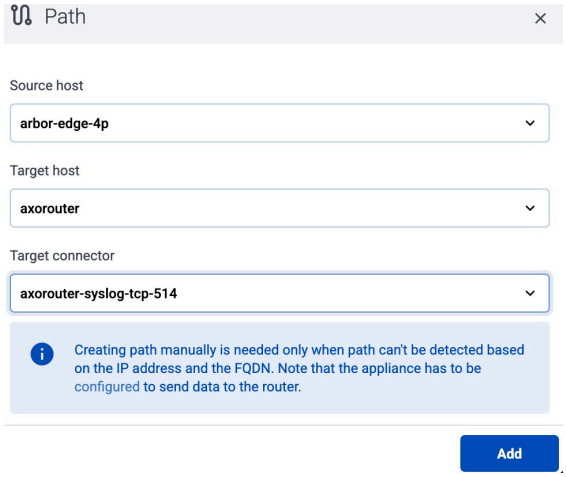
3. (Optional) Add custom labels as needed.

> **CAUTION:**
>
> AxoConsole can automatically raise an alert if a source that used to send data regularly stops sending data. To enable this alert for a source, add the `traffic_alerts:true` label to the source.

4. Select **Add**.

5. (Optional) **Add Path** manually. That's needed only when AxoConsole can't detect the path based on the IP address and the FQDN, and you haven't yet configured the source to send data to the router. If you've added the source from the **Detected** list, you can skip this step, as the path is created automatically.

> **Create a path between the source and the AxoRouter instance.**
>
> If you've added the source from the **Detected** list, you can skip this step, as the path is created automatically.
>
> 1. Select **Topology > Add Item > Path**.



2. If not set automatically, select the beginning of the path in the **Source host** field.
3. Select the target router or aggregator this source is sending its data to in the **Target host** field, for example, `axorouter`.
4. Select the **Target connector**. The connector determines how the destination receives the data (for example, using which protocol or port).
5. Select **Add**. The new path appears on the **Topology** page.



Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`dell`
`product`	`meta.product`	`sonicwall`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`dell:sonicwall`	`netfw`

Tested with: Dell SonicWall Add-on for Splunk technical add-on

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `SONIC_FIREWALL`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.44 Splunk

Route data from Splunk Universal Forwarders and Heavy Forwarders into Axoflow.

7.12.44.1 Edge Processor

Send data from Splunk Edge Processors to the Splunk HEC Server connector of AxoRouter.

Axoflow can receive data from Splunk Edge Processors using its Splunk HEC Server implementation.

1. Create a **Splunk HEC Server** connector rule in AxoConsole. For details, see Splunk HEC Server.
2. Configure your Splunk Edge Processors to send data using their `splunkhec` exporter to AxoRouter.

7.12.44.2 Heavy Forwarder

Forward data from Splunk Heavy Forwarders to AxoRouter, and check how AxoRouter classifies the forwarded data.

Heavy Forwarder: Receive data from Splunk.

This page describes how to configure your Splunk Heavy Forwarders to send data to AxoRouter.

Forwarding considerations

The Axoflow Forwarder for Splunk App works beside an existing Splunk configuration by creating an `axoflow` server group. Messages get cloned to the `axoflow` group separately during a transformation process.

The `axoflow` server group *should not* be part of the `defaultGroup` in the `[tcpout]` stanza otherwise all messages will be sent to AxoRouter twice. If the `defaultGroup` is not configured already, you can either:

- Disable forwarding completely.

```
[tcpout]
defaultGroup = NoForward
```

- Use a specific default group. For example, use `default-autolb-group`, as servers added through the UI will be part of it by default.

```
[tcpout]
defaultGroup = default-autolb-group
```

Prerequisites

- Once you made sure `defaultGroup` is configured correctly, download the latest release of Axoflow Forwarder for Splunk App, you'll need to install it on your Heavy Forwarders. The releases are published on the Axoflow Forwarder for Splunk App GitHub page: find the **Assets** section of the latest release, and download the `axoflow_forwarder-<version-number>.spl` file.

- You know the IP address the AxoRouter. To find it:

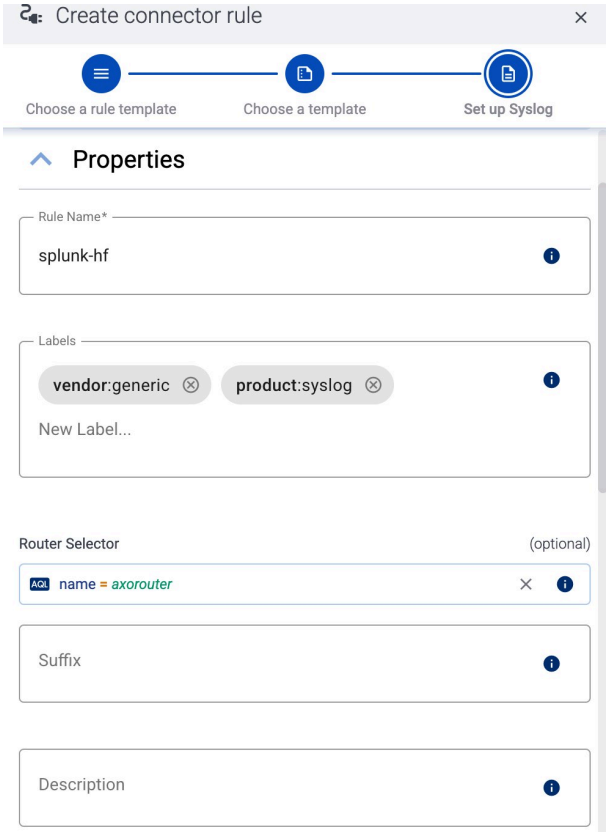
  1. Open the AxoConsole.
  2. Select the **Routers** or the **Topology** page.
  3. Select on AxoRouter instance that is going to receive the logs.
  4. Check the **Networks > Address** field.

Steps

To configure your Splunk Heavy Forwarders to send data to AxoRouter. Complete the following steps.

1. Create a new Syslog Connector rule with the following parameters:

   1. Select **Routers > Add Rule > Syslog > Custom**
   2. Enter `splunk-hf` into the **Rule Name** field.

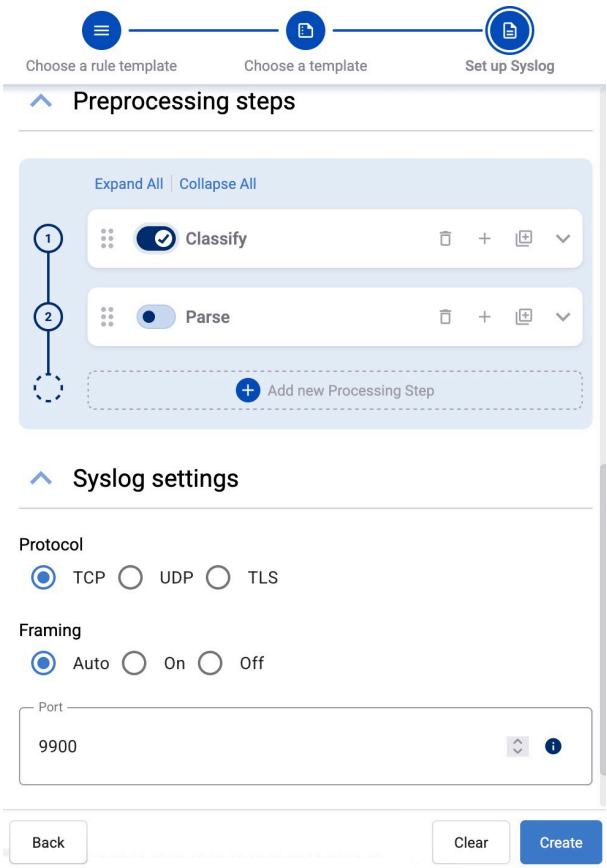


   3. Set the **Router Selector** so it matches the AxoRouter instances where your Splunk Heavy Forwarders will be forwarding their data. If you leave the **Router Selector** field empty, the rule will match all AxoRouters.

      You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

- If you leave the router selector empty, it matches every AxoRouter instance.
- To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.
- If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

4. In the **Preprocessing steps** section, enable **Classify**.



5. In the **Syslog settings** section:
    1. Select the **TCP** protocol.
    2. Select **NUL terminated** framing.
    3. Enter **9900** into the **Port** field.
6. Select **Add**.

2. Install the Axoflow Forwarder for Splunk App you've downloaded from the GitHub Releases Page on your Splunk Heavy Forwarders.
    1. Configure name resolution for the `axorouter` host by completing one of the following:
        - Add `axorouter` to the `/etc/hosts` file to resolve to the IP address of your AxoRouter instance where this host is sending data.

- Alternatively, you can add the following snippet to your `/opt/splunk/etc/system/local/outputs.conf` file:

```
[tcpout:axoflow]
server = <AXOROUTER_IP1>:9900, <AXOROUTER_IP2>:9900

# configure maxQueueSize to allow for a temporary in-memory buffer if
the destination is slow or unavailable
# maxQueueSize = 100MB
# configure a persistentQueueSize to allow for data to be queued on
disk if the destination is slow or unavailable
# persistentQueueSize = 1GB
```

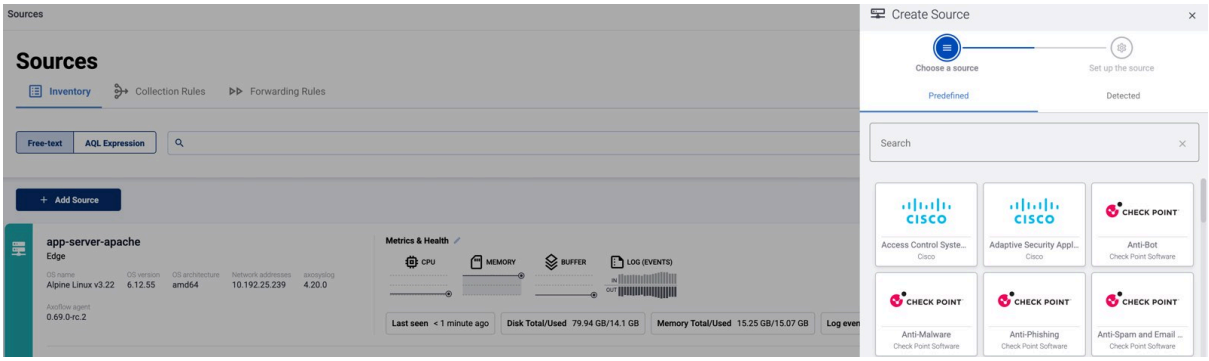
  Note that if you set multiple AxoRouters in the `server` field, the forwarder will load-balance among them.

  Configure either in-memory (`maxQueueSize`) or on-disk (`persistentQueueSize`) queueing to avoid data loss in case the destination is slow or unavailable.

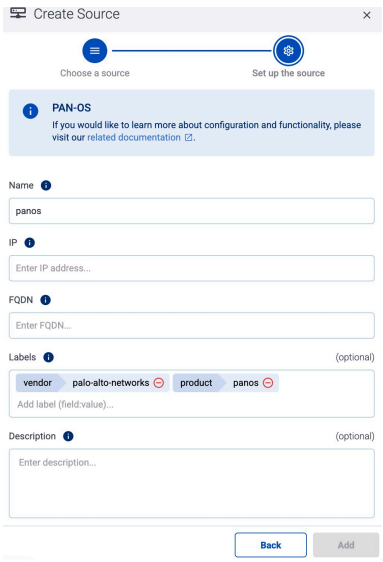
2. Install the Axoflow Forwarder for Splunk App using the Splunk UI.
3. Restart `splunkd` .

3. Add the source to AxoConsole.

1. Open the AxoConsole and select **Topology**.
2. Select **Add Item > Source**.



- If the source is actively sending data to an AxoRouter instance, select **Detected**, then select your source(s) and select **Add Selected Source(s)**.
- Otherwise, select the vendor and product corresponding to your source from the **Predefined** sources, then enter the parameters of the source, like **IP address** and **FQDN**.



> **Note**
>
> **During log tapping, you can add hosts that are actively sending data to an AxoRouter instance by clicking Register source.**

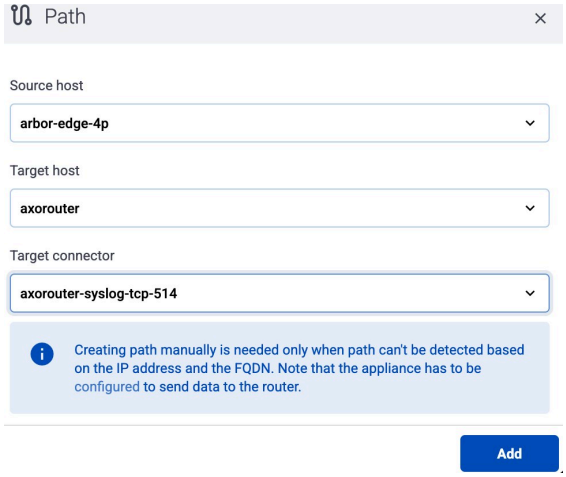
3. (Optional) Add custom labels as needed.

> **CAUTION:**
>
> AxoConsole can automatically raise an alert if a source that used to send data regularly stops sending data. To enable this alert for a source, add the `traffic_alerts:true` label to the source.

4. Select **Add**.

5. (Optional) **Add Path** manually. That's needed only when AxoConsole can't detect the path based on the IP address and the FQDN, and you haven't yet configured the source to send data to the router. If you've added the source from the **Detected** list, you can skip this step, as the path is created automatically.

> **Create a path between the source and the AxoRouter instance.**
>
> If you've added the source from the **Detected** list, you can skip this step, as the path is created automatically.
>
> 1. Select **Topology > Add Item > Path**.



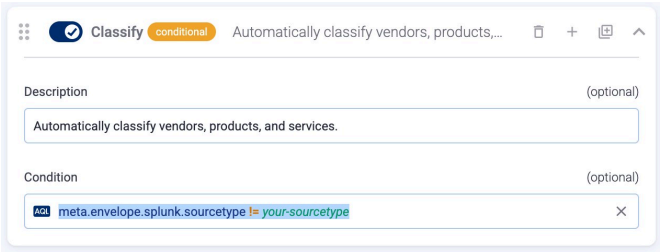
2. If not set automatically, select the beginning of the path in the **Source host** field.
3. Select the target router or aggregator this source is sending its data to in the **Target host** field, for example, `axorouter`.
4. Select the **Target connector**. The connector determines how the destination receives the data (for example, using which protocol or port).
5. Select **Add**. The new path appears on the **Topology** page.



Classification on AxoRouter

Note that when AxoRouter receives logs from Splunk HF using the Axoflow Forwarder for Splunk App:

- If classification is disabled on AxoRouter, the original source, sourcetype, index, timestamp and host metadata are preserved.
- If classification is enabled, then a successful classification overrides the sourcetype and the index, while the timestamp and host are overwritten if the original message contains these more specific information.
- If you want to disable the automatic classification on AxoRouter for certain messages received from the Axoflow Forwarder for Splunk App, add a Condition to the **Classify** processing step in the related flow, so the Condition won't match the sourcetype. For example: `meta.envelope.splunk.sourcetype != your-sourcetype`


Classify
conditional
Automatically classify vendors, products,...
Description
(optional)
Automatically classify vendors, products, and services.
Condition
(optional)
AQL
meta.envelope.splunk.sourcetype != your-sourcetype

7.12.44.3 OpenTelemetry Collector

Send data from the Splunk Distribution of the OpenTelemetry Collector to the Splunk HEC Server connector of AxoRouter.

Axoflow can receive data from Splunk OpenTelemetry Collectors using its Splunk HEC Server implementation.

1. Create a **Splunk HEC Server** connector rule in AxoConsole. For details, see Splunk HEC Server.
2. Configure your Splunk OpenTelemetry Collectors to send data using their `splunkhec` exporter to AxoRouter.

7.12.44.4 SC4S

Send data from Splunk Connect for Syslog (SC4S) to the Splunk HEC Server connector of AxoRouter.

Axoflow can receive data from Splunk Connect for Syslog (SC4S) using its Splunk HEC Server implementation.

1. Create a **Splunk HEC Server** connector rule in AxoConsole. For details, see Splunk HEC Server.
2. Configure your Splunk Connect for Syslog (SC4S) deployments to send data using their HEC destination to AxoRouter.

Note that you can instrument SC4S deployments to send metrics to AxoConsole. For details, see Splunk Connect for Syslog (SC4S).

7.12.45 Superna

Send logs from Superna Eyeglass to Axoflow for processing and routing.

7.12.45.1 Eyeglass

Configure Superna Eyeglass to send logs to Axoflow, and look up the labels that Axoflow adds.

Eyeglass: Manages and automates data protection, DR, and reporting for PowerScale environments.

The following sections show you how to configure Superna Eyeglass to send their log data to Axoflow.

> **CAUTION:**
>
> Make sure to set data forwarding on your appliances/servers as described in this guide. Different settings like alternate message formats or ports might be valid, but can result in data loss or incorrect parsing.

Prerequisites

- You have administrative access to Superna Eyeglass.
- You have an AxoRouter deployed and configured with a webhook connector. This device is going to receive the data from Superna Eyeglass.
- You know the IP address the AxoRouter. To find it:

  1. Open the AxoConsole.
  2. Select the **Routers** or the **Topology** page.
  3. Select on AxoRouter instance that is going to receive the logs.
  4. Check the **Networks > Address** field.

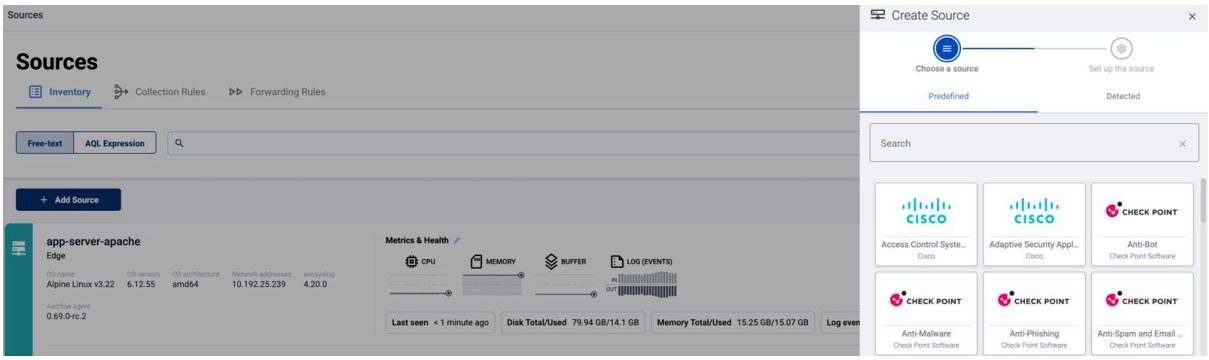
Steps

Note: The steps involving the Superna Eyeglass user interface are just for your convenience, for details, see the official documentation.

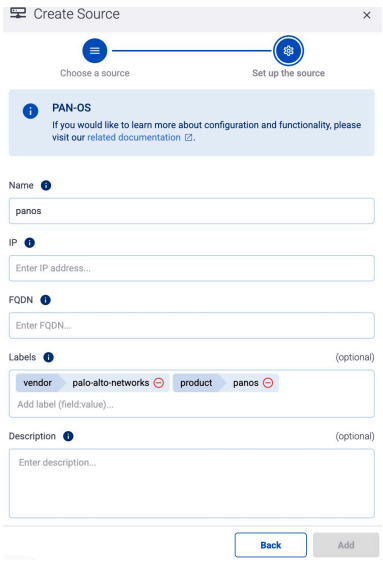
1. Log in to Ransomware Defender and open the Zero Trust menu.

2. Click the plus sign to add a webhook target.

3. Set the parameters of the webhook.
    - **Name**: Enter a name for the webhook, for example, `Axoflow`.
    - **URL**: Enter the URL of the webhook connector of the AxoRouter instance where you want to post messages.
    - **Event Severity Filter**: Select the severities of the events that you want to forward to the webhook.
    - **Lifecycle filter**: Select the lifecycle changes that trigger a post message to the webhook.
4. Click **Save**, then the **Test webhooks** button. This will send a post message with a sample payload.
5. Add the source to AxoConsole.
    1. Open the AxoConsole and select **Topology**.
    2. Select **Add Item > Source**.



    - If the source is actively sending data to an AxoRouter instance, select **Detected**, then select your source(s) and select **Add Selected Source(s)**.
    - Otherwise, select the vendor and product corresponding to your source from the **Predefined** sources, then enter the parameters of the source, like **IP address** and **FQDN**.



> ## Note
>
> **During log tapping, you can add hosts that are actively sending data to an AxoRouter instance by clicking Register source.**

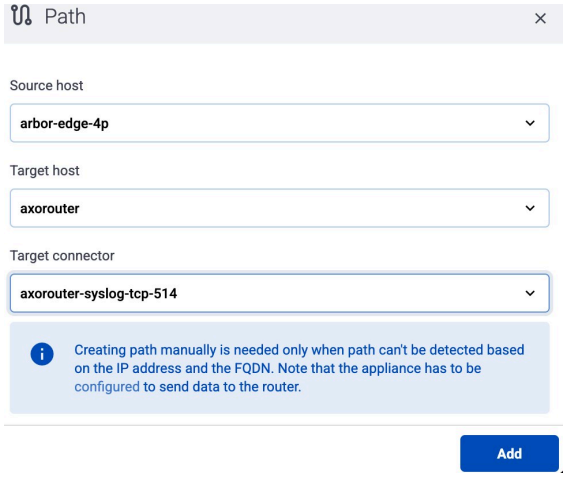
3. (Optional) Add custom labels as needed.

> **CAUTION:**
>
> AxoConsole can automatically raise an alert if a source that used to send data regularly stops sending data. To enable this alert for a source, add the `traffic_alerts:true` label to the source.

4. Select **Add**.

5. (Optional) **Add Path** manually. That's needed only when AxoConsole can't detect the path based on the IP address and the FQDN, and you haven't yet configured the source to send data to the router. If you've added the source from the **Detected** list, you can skip this step, as the path is created automatically.

> **Create a path between the source and the AxoRouter instance.**
>
> If you've added the source from the **Detected** list, you can skip this step, as the path is created automatically.
>
> 1. Select **Topology > Add Item > Path**.



2. If not set automatically, select the beginning of the path in the **Source host** field.
3. Select the target router or aggregator this source is sending its data to in the **Target host** field, for example, `axorouter`.
4. Select the **Target connector**. The connector determines how the destination receives the data (for example, using which protocol or port).
5. Select **Add**. The new path appears on the **Topology** page.



Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`superna`
`product`	`meta.product`	`eyeglass`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`superna:eyeglass`	`main`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `SUPERNA_EYEGLASS` .

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog` .

7.12.46 syslog-ng

Send logs from syslog-ng to AxoRouter as a generic syslog source, or onboard the host to get detailed metrics about it.

By default, Axoflow treats syslog-ng sources as a generic syslog source.

- The easiest way to send data from syslog-ng to Axoflow is to configure it to send data to an AxoRouter instance using the syslog protocol.
- If you're using syslog-ng Open Source Edition version 4.4 or newer, use the `syslog-ng-otlp()` driver to send data to AxoRouter using the OpenTelemetry Protocol.

Note that even if syslog-ng is acting as a relay (receiving data from other clients and forwarding them to AxoRouter), on the **Topology** page it will be displayed as a data source.

Prerequisites

- You have administrative access to the device running syslog-ng.
- The date, time, and time zone are correctly set on the appliance.
- You have an AxoRouter deployed and configured with a Syslog connector that has parsing and classification enabled (by default, every AxoRouter has such connectors). This device is going to receive the data from the source.
- You know the IP address the AxoRouter. To find it:

  1. Open the AxoConsole.
  2. Select the **Routers** or the **Topology** page.
  3. Select on AxoRouter instance that is going to receive the logs.
  4. Check the **Networks > Address** field.

> **Note**
>
> **To receive more detailed metrics about the data processed by syslog-ng, consider onboarding the host into Axoflow and instrumenting its configuration.**

7.12.47 Tanium

Collect data from the Tanium Platform and forward it to Axoflow for processing.

7.12.47.1 Tanium Platform

Send Tanium Platform logs to Axoflow, and look up the labels that Axoflow adds to its data.

Tanium Platform: Endpoint management and security software that provides organizations with visibility and control across their IT environments.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

When configuring Tanium Platform, make sure to:

- Enable TCP octet framing, and
- Enable RFC5424 output format

For details, see the Tanium Appliance documentation.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`tanium`
`product`	`meta.product`	`platform`
`service`	`meta.service.name`	`Tanium`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

source	sourcetype	index
Depends on the log message, always begins with `tanium:question:`, for example, `tanium:question:tanium_droid`	`tanium`, `tanium:audit`, `tanium:inventory`, `tanium:hardware:cpu`, `tanium:endpoint:process`, `tanium:endpoint:services`, `tanium:deploy:deploymeantime`, `tanium:change:endpoint`, `tanium:report:vulnerability`, `tanium:report:vulnerability`, `tanium:discover:report`, `tanium:malware:attack`, `tanium:updates`, `tanium:patch:patchmeantime`, `tanium:ids:netconns`, `tanium:report:vulnerability`, `tanium:report:vulnerability`, `tanium:discover:lost`, `tanium:discover:managed`, `tanium:discover:unmanaged`, `tanium:endpoint:dns:stream`, `tanium:endpoint:securityevent:stream`, `tanium:endpoint:library:stream`, `tanium:endpoint:processes:stream`, `tanium:endpoint:netconn:stream`, `tanium:endpoint:netdisco:stream`, `tanium:endpoint:netaccept:stream`, `tanium:endpoint:filecreate:stream`, `tanium:endpoint:filewrite:stream`, `tanium:endpoint:fileread:stream`, `tanium:endpoint:fileopen:stream`, `tanium:endpoint:filemove:stream`, `tanium:endpoint:filedelete:stream`, `tanium:endpoint:filepermchange:stream`, `tanium:endpoint:regcreate:stream`, `tanium:endpoint:regset:stream`, `tanium:endpoint:regdelete:stream`, `tanium:detect:signals`, `tanium:detect:openioc`, `tanium:detect:yara`, `tanium:detect:stix`	`tanium`

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `TANIUM_QUESTION`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.48 Tenable

Collect the audit log of Tenable One Vulnerability Management.

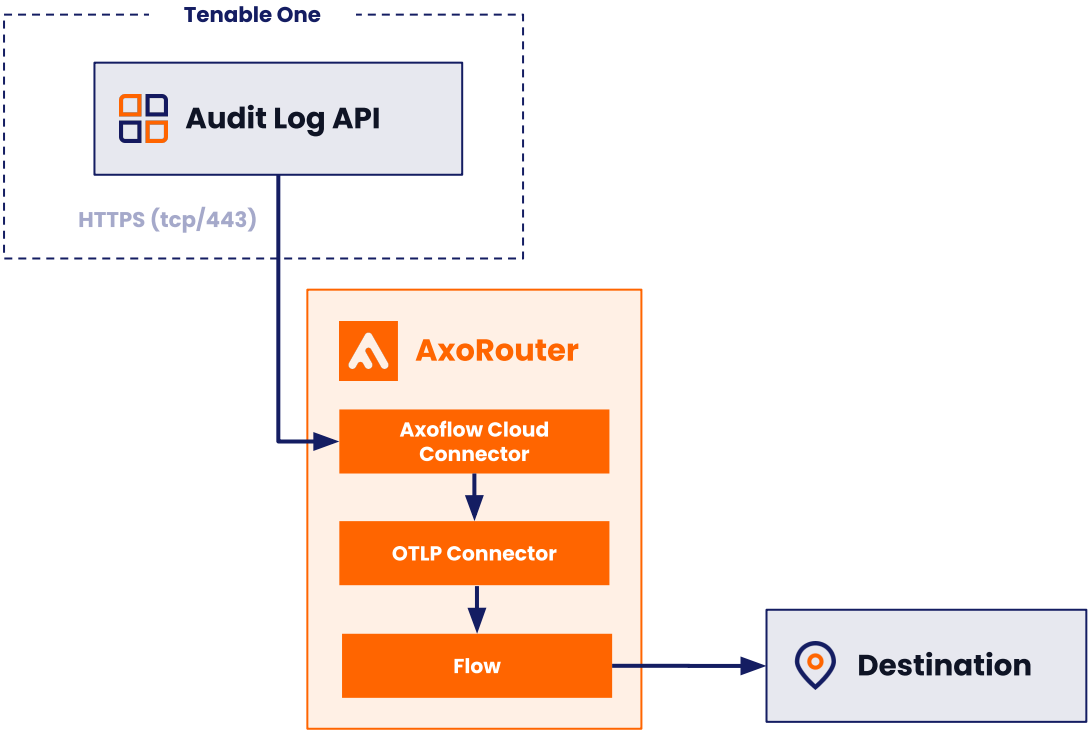
7.12.48.1 Tenable One Vulnerability Management

Collect the audit log of Tenable One Vulnerability Management with a Cloud Connector, and look up the labels Axoflow adds.

Tenable One Vulnerability Management: Cloud-based vulnerability management platform whose audit log API records user and configuration activity.

Axoflow can collect the audit log of your Tenable One Vulnerability Management deployment with an Axoflow Cloud Connector. At a high level, the process looks like this:

- Create a **Connector Rule** on the AxoRouter that deploys an Axoflow Cloud Connector and collects the data from the Tenable audit log API. Axoflow Cloud Connector is a simple container that runs on an AxoRouter instance.
- Configure a Flow on AxoConsole that processes and routes the collected data to your destination (for example, Splunk or another SIEM).



Prerequisites

- An AxoRouter instance that:
  - Was provisioned with **Enable Cloudconnectors** turned on.

- Can reach the Tenable Vulnerability Management API (by default, cloud.tenable.com) on TCP port 443. Adjust your firewall rules and the access rules of the data source as needed.
- You know the IP address the AxoRouter. To find it:

  1. Open the AxoConsole.
  2. Select the **Routers** or the **Topology** page.
  3. Select on AxoRouter instance that is going to receive the logs.
  4. Check the **Networks > Address** field.

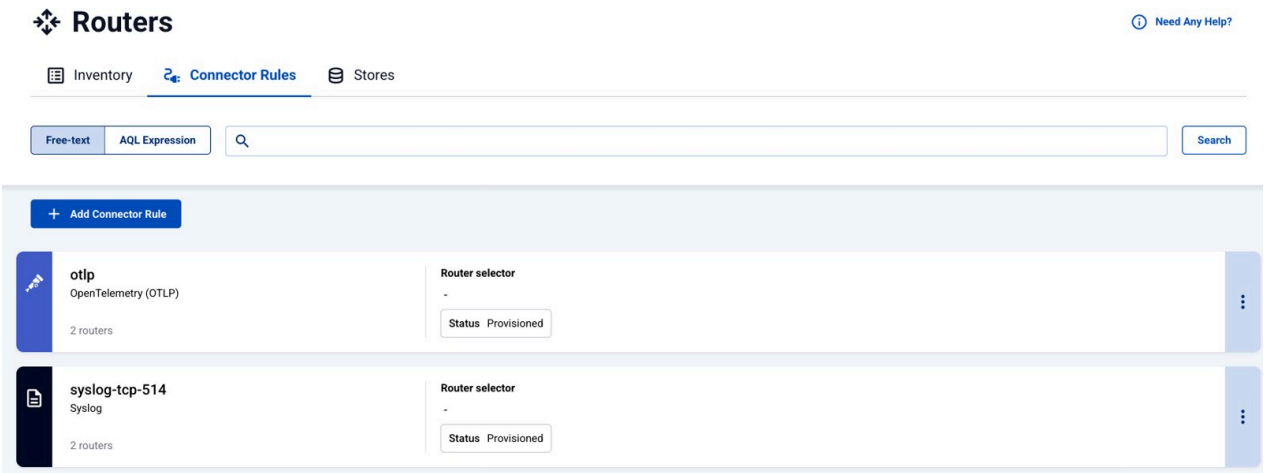
- A Tenable Vulnerability Management account with an active subscription.
- An API key pair (an access key and a secret key) generated for a user who can read the audit log of your container.

> **Note**
>
> **The Tenable audit log API retains the events of the last 30 days only. Axoflow Cloud Connector cannot collect events that are older than that.**

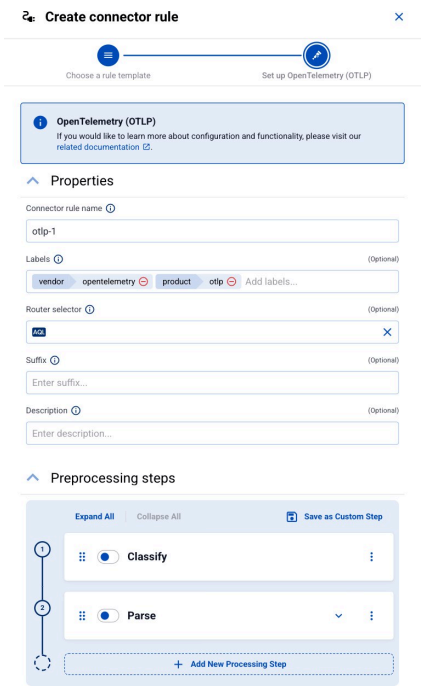
Configure Axoflow Cloud Connector on AxoRouter

To collect data from Tenable One Vulnerability Management, complete the following steps.

1. Select **Routers > Connector Rules > Add Connector Rule**. (Alternatively, you can select **Add Connector > Add a connector rule** on the **Connectors** page of an AxoRouter host.)



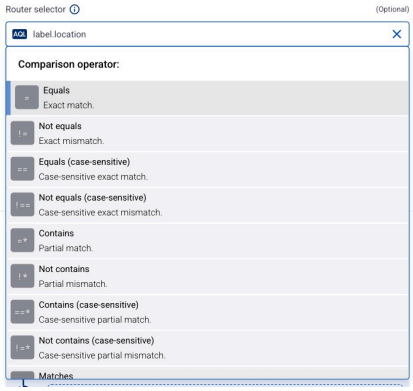
2. Select **Vulnerability Management** (Tenable).
3. Enter a name for the connector rule into the **Connector rule name** field.



4. (Optional) Add labels to the connector rule. You will be able to use these labels in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps. For details about the message schema and the related connector-specific fields, see the `meta.connector` object.

5. Set the **Router selector** for the connector rule. The selector determines which AxoRouter instances get a connector from this connector rule.

   > **CAUTION:**
   >
   > Each connector rule for Tenable One Vulnerability Management must match exactly one AxoRouter. Use the `name` or `guid` AQL selector to make sure it matches only one AxoRouter. Otherwise several Axoflow Cloud Connector instances collect the same data, and Axoflow receives every message multiple times.



   You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

   - If you leave the router selector empty, it matches every AxoRouter instance.

- To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.
- If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

6. (Optional) Enter a **Suffix** for the connector rule. AxoConsole uses the suffix in the name of the connector instances that it creates on the AxoRouter hosts. The name of the connector instance follows the `<name-of-AxoRouter>-<suffix>` schema. For example, if the name of a matching AxoRouter instance is `axorouter-1`, and the suffix of the rule is `custom`, the name of the connector on that AxoRouter is `axorouter-1-custom`.

   If the **Suffix** field is empty, AxoConsole uses the name of the connector rule instead. For example, if the name of the connector rule is `otlp-1`, the name of the connector is `axorouter-1-otlp-1`.

7. (Optional) Enter a description for the rule.

8. Configure how Axoflow Cloud Connector reaches the Tenable audit log API.

   - **Access key** and **Secret key**: The Tenable API key pair that Axoflow Cloud Connector uses to authenticate.
   - **Endpoint**: (Optional) The base URL of the Tenable Vulnerability Management API. Default value: `https://cloud.tenable.com`

9. (Optional) Select **More options** to change the advanced options of the connector, for example, how often it polls the API.

10. Select **Add**. The Axoflow Cloud Connector starts polling the Tenable audit log API and forwarding the collected events to the AxoRouter instance.

11. Verify that the Axoflow Cloud Connector is running.

    1. Select **Routers**, then select the AxoRouter you added the Axoflow Cloud Connector to.
    2. Select **Services** and check that **axoflow-cloudconnector.service** is running without errors.

12. Create a Flow to route the data from the AxoRouter instance to a destination. You can use the Labels of this source to select its messages.

Advanced options

Select **More options** on the connector rule to set the following options.

- **Poll interval**: How often Axoflow Cloud Connector queries the audit log API. Default value: `5m`
- **Page size**: The number of events that Axoflow Cloud Connector requests per API call. Minimum value: `1`. Maximum value: `5000`. Default value: `1000`
- **Max records per poll**: Caps the number of events that a single poll collects. Axoflow Cloud Connector collects the remaining events during the following polls. Minimum value: `1`. Default value: `5000`
- **Initial lookback**: How far back Axoflow Cloud Connector collects events when no checkpoint is stored yet. Set it to `0` to collect only the events that arrive after the connector starts. Default value: `24h`

  Axoflow Cloud Connector checkpoints the position of the poll cursor, so after a restart it resumes the collection where the last poll stopped. The **Initial lookback** option applies only when there is no stored checkpoint. Since the Tenable audit log retains 30 days of events, setting a longer period doesn't collect more data.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`tenable`
`product`	`meta.product`	`vulnerability-management`
`service`	`meta.service.name`	`audit`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

7.12.49 Thales

Send logs from the Thales Vormetric Data Security Platform to Axoflow.

7.12.49.1 Vormetric Data Security Platform

Send Thales Vormetric Data Security Platform logs to Axoflow, and look up the labels that Axoflow adds to its data.

Vormetric Data Security Platform: Provides data encryption, key management, and access controls across cloud and on-premise environments.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`thales`
`product`	`meta.product`	`vormetric`
`service`	`meta.service.name`	`vee-FS`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index

```
thales:vormetric    netauth
```

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `VORMETRIC` .

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog` .

7.12.50 Trellix

Send logs from Trellix ePO, Endpoint Security (HX), Email Threat Prevention, IPS, CMS, and MPS to Axoflow.

7.12.50.1 Central Management System (CMS)

Send Trellix Central Management System (CMS) logs to Axoflow, and look up the labels that Axoflow adds to its data.

Central Management System (CMS): Centralized policy and configuration management platform for Trellix security products.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`trellix`
`product`	`meta.product`	`cms`
`service`	`meta.service.name`	`fenotify`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	source	index

```
trellix:cms   trellix:cms   netops
```

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `FIREEYE_CMS` .

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog` .

7.12.50.2 Email Threat Prevention (ETP)

Send Trellix Email Threat Prevention (ETP) logs to Axoflow, and look up the labels that Axoflow adds to its data.

Email Threat Prevention (ETP): Analyzes and filters email traffic to block phishing, malware, and targeted email-based threats.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`trellix`
`product`	`meta.product`	`etp`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`fe_etp`	`fireeye`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `FIREEYE_ETP`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.50.3 Endpoint Security (HX)

Send Trellix Endpoint Security (HX) logs to Axoflow, and look up the labels that Axoflow adds to its data.

Endpoint Security (HX): Detects and responds to advanced threats on endpoints using behavior-based analysis and threat intel.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Note that the device can be configured to send logs formatted as JSON or CEF. AxoRouter can automatically parse all flavors.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`trellix`
`product`	`meta.product`	`hx`
`service`	`meta.service.name`	`cef` , `fenotify`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`hx_json`	`fireeye`

`fe_json`	`fireeye`
`hx_cef_syslog`	`fireeye`

Tested with: FireEye Add-on for Splunk Enterprise

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `FIREEYE_HX` .

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog` .

Earlier name/vendor

FireEye Endpoint Security (HX)

7.12.50.4 ePolicy Orchestrator (EPO)

Send Trellix Email Threat Prevention (ETP) logs to Axoflow, and look up the labels that Axoflow adds to its data.

ePolicy Orchestrator (EPO): Analyzes and filters email traffic to block phishing, malware, and targeted email-based threats.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`trellix`
`product`	`meta.product`	`epo`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index	source
`mcafee:epo:syslog`	`epav`	`trellix_endpoint_security`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `MCAFEE_EPO`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

Earlier name/vendor

McAfee ePolicy Orchestrator (EPO)

7.12.50.5 Intrusion Prevention System

Send Trellix Intrusion Prevention System logs to Axoflow, and look up the labels that Axoflow adds to its data.

Intrusion Prevention System: Trellix Intrusion Prevention System is an NDR-ready, next-generation IPS that detects and blocks sophisticated malware threats across the network.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`trellix`
`product`	`meta.product`	`ips`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`mcafee:nsp`	`netids`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `MCAFEE_NSP` .

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog` .

Earlier name/vendor

- McAfee Network Security Platform

7.12.50.6 MPS

Send Trellix MPS logs to Axoflow, and look up the labels that Axoflow adds to its data.

MPS: Appliance for detecting and blocking advanced threats through inline malware inspection.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`trellix`
`product`	`meta.product`	`mps`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	source	index
`trellix:mps`	`trellix:mps`	`netops`

7.12.51 Trend Micro

Collect logs from Trend Micro Deep Security Agent and send them to Axoflow.

7.12.51.1 Deep Security Agent

Send Trend Micro Deep Security Agent logs to Axoflow, and look up the labels that Axoflow adds to its data.

Deep Security Agent: Provides anti-malware, intrusion prevention, and log inspection for cloud and on-prem servers.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`trend-micro`
`product`	`meta.product`	`deep-security-agent`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`deepsecurity`	`epintel`

`deepsecurity-system_events`	`epintel`
`deepsecurity-intrusion_prevention`	`epintel`
`deepsecurity-firewall`	`epintel`
`deepsecurity-antimalware`	`epintel`
`deepsecurity-integrity_monitoring`	`epintel`
`deepsecurity-log_inspection`	`epintel`
`deepsecurity-web_reputation`	`epintel`
`deepsecurity-app_control`	`epintel`
`deepsecurity-system_events`	`epintel`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `TRENDMICRO_DEEP_SECURITY` .

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog` .

7.12.52 Ubiquiti

Send logs from Ubiquiti UniFi network devices to Axoflow for processing.

7.12.52.1 Unifi

Send Ubiquiti Unifi logs to Axoflow, and look up the labels that Axoflow adds to its data.

Unifi: Manages network devices including routers, switches, and access points with centralized control.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`ubiquiti`
`product`	`meta.product`	`unifi`
`service`	`meta.service.name`	`crond` , `logread` , `kernel` , `mcad` , `switch` , `UDMPRO`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index

`ubnt`	`netops`
`ubnt:cef`	`netops`
`ubnt:dhcp`	`netops`
`ubnt:dnsmasq`	`netops`
`ubnt:edgeswitch`	`netops`
`ubnt:hostapd`	`netops`
`ubnt:link`	`netops`
`ubnt:mcad`	`netops`
`ubnt:sudo`	`netops`
`ubnt:wireless`	`netops`
`ubnt:fw`	`netfw`
`ubnt:fw:cef`	`netfw`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `UBIQUITI_SWITCH`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.53 Varonis

Send logs from Varonis DatAdvantage to Axoflow for processing and routing.

7.12.53.1 DatAdvantage

Send Varonis DatAdvantage logs to Axoflow, and look up the labels that Axoflow adds to its data.

DatAdvantage: Monitors data access and permissions to detect insider threats and automate compliance reporting.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`varonis`
`product`	`meta.product`	`datadvantage`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`varonis:ta`	`main`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `VARONIS`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.54 Vectra AI

Send detections from the Vectra AI Platform (formerly Vectra Cognito) to Axoflow.

Earlier name/vendor

Vectra Cognito

7.12.54.1 AI Platform

Send Vectra AI Platform logs to Axoflow, and look up the labels that Axoflow adds to its data.

AI Platform:

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`vectra`
`product`	`meta.product`	`x-series`
`service`	`meta.service.name`	`vectra_cef`, `vectra_cef_account_detection`, `vectra_cef_audit`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`vectra:cognito:detect`	`main`
`vectra:cognito:accountdetect`	`main`
`vectra:cognito:accountscoring`	`main`

`vectra:cognito:audit`	`main`
`vectra:cognito:campaigns`	`main`
`vectra:cognito:health`	`main`
`vectra:cognito:hostscoring`	`main`
`vectra:cognito:accountlockdown`	`main`

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `VECTRA_DETECT`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

Earlier name/vendor

- Cognito Detect
- Vectra Detect

7.12.55 Zscaler appliances

Send logs from Zscaler Nanolog Streaming Service (ZIA) and Log Streaming Service (ZPA) to Axoflow.

7.12.55.1 Zscaler Nanolog Streaming Service

Send Zscaler Internet Access logs to Axoflow, and look up the labels that Axoflow adds to its data.

Zscaler Nanolog Streaming Service: Cloud-based secure internet gateway that inspects traffic for threats and enforces policies.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`zscaler`
`product`	`meta.product`	`nss`
`service`	`meta.service.name`	`ZscalerNSS`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index

`zscalernss-alerts`	`netops`
`zscalernss-tunnel`	`netops`
`zscalernss-web`	`netproxy`
`zscalernss-web:leef`	`netproxy`

Tested with: Zscaler Technical Add-On for Splunk

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `ZSCALER_INTERNET_ACCESS`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

7.12.55.2 Zscaler Log Streaming Service

Send Zscaler Private Access logs to Axoflow, and look up the labels that Axoflow adds to its data.

Zscaler Log Streaming Service: Provides secure remote access to internal apps without exposing them to the public internet.

To onboard such a source to Axoflow, complete the generic appliance onboarding steps.

Labels

Axoflow automatically adds the following labels to data collected from this source:

Analytics label	Message field	Value
`vendor`	`meta.vendor`	`zscaler`
`product`	`meta.product`	`lss`

You can use the labels as:

- labels in **Customize Graph** on the Analytics page,
- in the **Filter labels** field during log tapping.

You can use the message fields

- in Flow Processing steps, for example, in the **Query** field of **Select Messages** steps,
- in AQL expressions in the search bars.

Sending data to Splunk

When sending the data collected from this source to Splunk, Axoflow uses the following sourcetype and index settings:

sourcetype	index
`zscalerlss-zpa-app`	`netproxy`
`zscalerlss-zpa-audit`	`netproxy`
`zscalerlss-zpa-auth`	`netproxy`
`zscalerlss-zpa-bba`	`netproxy`

`zscalerlss-zpa-connector` `netproxy`

Tested with: Zscaler Technical Add-On for Splunk

If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).

Sending data to Google SecOps

When sending the data collected from this source to a *dynamic* Google SecOps destination, Axoflow sets the following log type: `ZSCALER_ZPA, ZSCALER_ZPA_AUDIT`.

Sending data to Microsoft Sentinel

When sending the data collected from this source to a Microsoft Sentinel destination, Axoflow normalizes the data and sends it to the following table: `Syslog`.

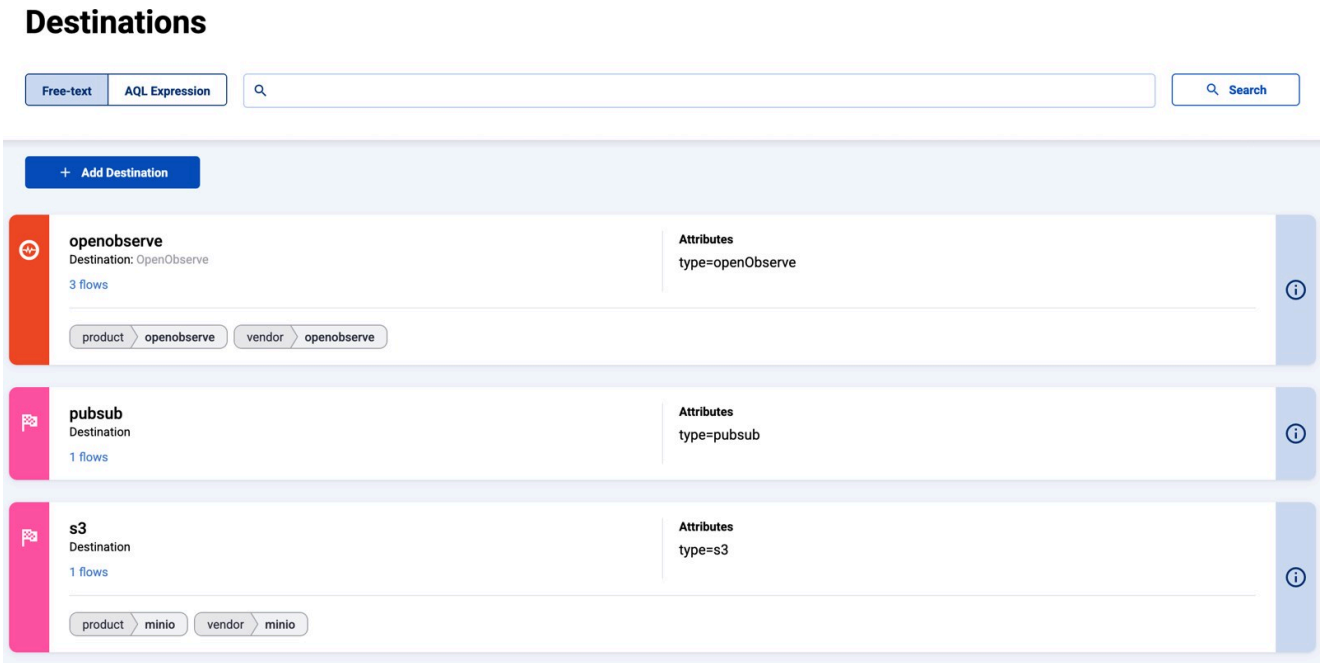
8 Destinations

Configure Axoflow to send your security data to SIEMs, object storage, observability platforms, and other destinations.

The following chapters show you how to configure Axoflow to send your data to specific destinations, like SIEMs or object storage solutions.

- You can find the list of existing destinations on the **Destinations** page.
- To modify an existing destination, select its name from the list, then select **Edit**.
- Stores and External Stores are also listed as destinations.
  - AxoStore stores logs locally on the AxoRouter that processed the messages.
  - External stores use a separately deployed AxoLake or an existing compatible ClickHouse cluster.
  - Logstore is a new store for local and external data storage based on Apache Parquet and Apache Avro. Logstore is now in early-access.

  To create a new store, see Create Store or Create External Store.
- To create a new destination, select **Add Destination**. See the section of the specific destination for details.
- To find and add a supported destination, you can also use the Integrations page.



8.1 Amazon

Configure Axoflow to write your log data into Amazon S3 buckets.

8.1.1 Amazon S3

Add an Amazon S3 destination to send your security data to an S3 bucket, and look up the message fields it uses.

Amazon S3: Scalable cloud storage service for storing and retrieving any amount of data via object storage.

To add an Amazon S3 destination to Axoflow, complete the following steps.

Prerequisites

- An existing S3 bucket configured for programmatic access, and the related `ACCESS_KEY` and `SECRET_KEY` of a user that can access it. The user needs to have the following permissions:
  - `s3:ListBucket`
  - `s3:ListBucketMultipartUploads`
  - `s3:AbortMultipartUpload`
  - `s3:ListMultipartUploadParts`
  - `s3:PutObject`

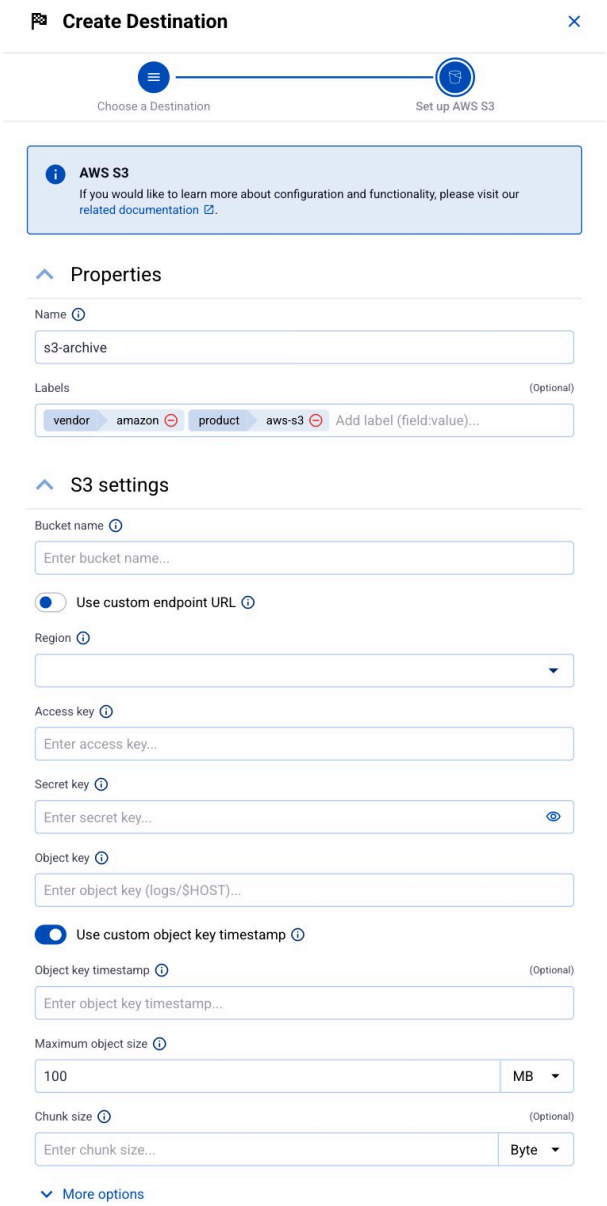
  If the default encryption of the bucket is SSE-KMS (`aws:kms`), the AWS role or user also needs the following permissions on the KMS key:
  - `kms:Decrypt` (For details on why the `kms:Decrypt` is mandatory, check this AWS Knowledge Center entry.)
  - `kms:Encrypt`
  - `kms:GenerateDataKey`

> **See a sample IAM policy**
>
> ```
> {
>     "Version": "2012-10-17",
>     "Statement": [
>         {
>             "Sid": "bucketLevel",
>             "Effect": "Allow",
>             "Action": [
>                 "s3:ListBucket",
>                 "s3:ListBucketMultipartUploads",
>                 "s3:AbortMultipartUpload",
>                 "s3:ListMultipartUploadParts"
>             ],
>             "Resource": "arn:aws:s3:::YOUR-BUCKET-NAME"
>         },
>         {
>             "Sid": "objectLevel",
>             "Effect": "Allow",
>             "Action": [
>                 "s3:PutObject"
>             ],
>             "Resource": "arn:aws:s3:::YOUR-BUCKET-NAME/*"
>         }
>     ]
> }
> ```

- To configure Axoflow, you'll need the bucket name, region (or URL), access key, and the secret key of the bucket.

Steps

1. Create a new destination.
    1. Open the AxoConsole.
    2. Select **Destinations > Add Destination > Destination**.
2. Configure the destination.
    1. Select **AWS S3**.
    2. Enter a name for the destination.



3. (Optional) Add custom labels to the destination.

4. Enter the name of the bucket you want to use into the **Bucket name** field.

5. Enter the region code of the bucket into the **Region** field (for example, `us-east-1`.), or select the **Use custom endpoint URL** option, and enter the URL of the endpoint into the **URL** field.

6. Enter the **Access key** and the **Secret key** for the account you want to use.

7. Enter the **Object key** (or key name), which uniquely identifies the object in an Amazon S3 bucket, for example: `my-logs/${HOSTNAME}/`.

   You can use AxoSyslog macros in this field.

1. (Optional) Set the **Object key timestamp**. By default, **Use custom object key timestamp** is enabled, so you can enter a custom template into the **Object key timestamp** field. To select one of the predefined formats instead (for example,

`${R_YEAR}/${R_MONTH}/${R_DAY}`), disable **Use custom object key timestamp**. For details on the available date-related macros, see the AxoSyslog documentation.

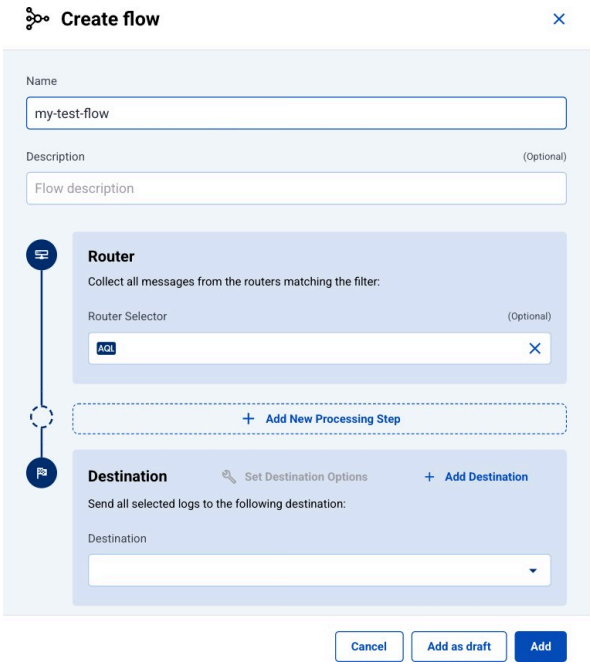
2. Set the maximal size of the S3 object in the **Maximum object size** field. If an object reaches this size, Axoflow appends an index ("-1", “-2”, ...) to the end of the object key and starts a new object after rotation. Default value: 100 MB.
3. (Optional) If needed, adjust the size of messages written to the S3 object in a batch using the **Chunk size** field.
4. (Optional) Select **More options** to set other options as needed for your environment.
   - **Message format**: Specifies the message format and normalization. Possible values:
     - **Elastic Common Schema (ECS)**: Normalize the data using the ECS schema.
     - **Open Cybersecurity Schema Framework (OCSF)**: Normalize the data using the OCSF schema.
     - **Parsed**: Send the parsed data without further normalization.
5. Select **Add**.

3. **Create a flow to connect the new destination to an AxoRouter instance.**
   1. Select **Flows**.
   2. Select **Add Flow > Flow**.

      To create a fallback flow, select **Add Flow > Fallback flow**.

   3. Enter a name for the flow, for example, `my-test-flow`.



4. In the **Router Selector** field, enter an expression that matches the routers you want to apply the flow to. To select a specific router, use a name selector, for example, `name = my-axorouter-hostname`.

   You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

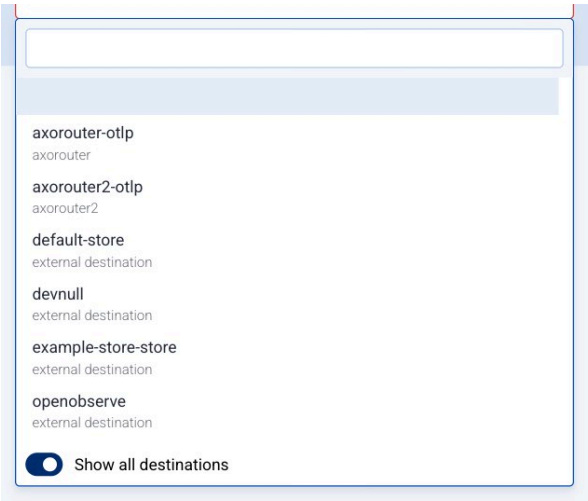
   - If you leave the router selector empty, it matches every AxoRouter instance.
   - To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.
   - If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

   > **Note**
   >
   > **You can configure multiple fallback flows, but only one fallback flow can apply to an AxoRouter (so the Router Selector of the fallback flows can't overlap).**

5. Select the **Destination** where you want to send your data. If you don't have any destination configured, you can select **Add Destination** in the destination section to create a new destination now. For details on the different destinations, see Destinations.

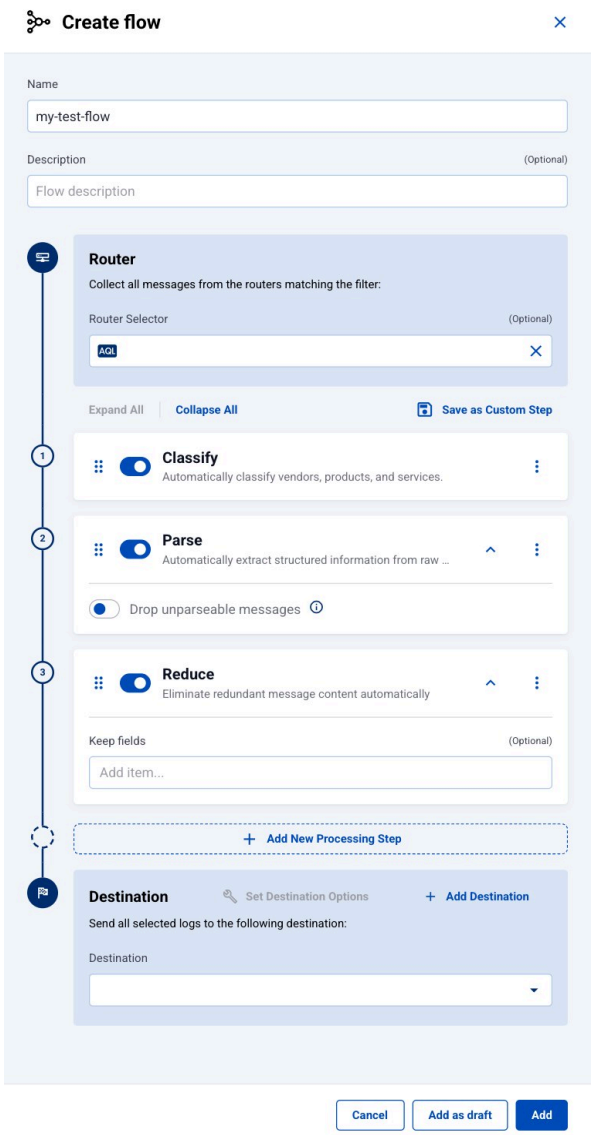
   - If you don't have any destination configured, see Destinations.

- If you've already created a store, it's automatically available as a destination. Note that the **Router Selector** of the flow must match only AxoRouters that have the selected store available, otherwise you'll get an error message.
- If you want to send data to another AxoRouter, enable the **Show all destinations** option, and select the connector of the AxoRouter where you want to send the data.



6. (Optional) To process the data transferred in the flow, select **Add New Processing Step**. For details, see Processing steps. For example:

   1. Add a **Classify**, a **Parse**, and a **Reduce** step, in that order, to automatically remove redundant and empty fields from your data.
   2. To select which messages are processed by the flow, add a **Select Messages** step, and enter a filter into the **AQL Expression** field. For example, to select only the messages received from Fortinet FortiGate firewalls, use the `meta.vendor = fortinet AND meta.product = fortigate` query.



7. Select **Add**, then confirm with **Add**. To save the flow without provisioning it to the routers yet, select **Add as draft** instead.

8. The new flow appears in the **Flows** list.



Related message fields

You can use the following message fields to modify messages sent to this destination using processing steps.

Troubleshooting

If the data doesn't arrive at the destination, see the data doesn't reach the destination. To find the errors that the destination returns, see the destination rejects the data. For other problems, see Troubleshooting Axoflow.

8.2 Apache Kafka

Support for an Apache Kafka destination is coming soon. Request it, or ask about the status of the integration.

Apache Kafka: Distributed event streaming platform used for building real-time data pipelines and stream processing.

If you'd like to send data from AxoRouter to this destination, contact our support team for details.

8.3 AxoStore

Send data to AxoStore, the local storage on your AxoRouter hosts, and access the stored data.

AxoStore is a lightweight, queryable, temporary storage option that stores data locally on the AxoRouter host. AxoConsole provides federated search for every AxoRouter that has AxoStore deployed, so you can simultaneously run queries on every AxoStore.

AxoStore is automatically available for all Axoflow customers, for each AxoRouter node, with 7 days retention, and up to 1TB of storage. For longer retention time or bigger storage, you need an extended AxoStore subscription. Contact us for the details.

You can retrieve selected data from AxoStore and (rehydrate) it to other destinations for further analysis as needed.

To send data to AxoStore, complete the following steps.

Prerequisites

- AxoStore is available only if you've deployed a AxoRouter instance with AxoStore enabled. For details, see Install AxoRouter on Linux.
- AxoStore is a local storage on the AxoRouter node. Only data received by the AxoRouter node can be sent to the AxoStore.
- By default, there are no stores defined. See Create Store.

Steps

Create a flow to route data received by AxoRouter to the local AxoStore destination. Note the following points:

- The amount of data you can store in AxoStore and its retention time is limited depending on your Axoflow subscription.
- By default, AxoStore is enabled on every AxoRouter instance. If you've disabled it on some instances, make sure that the **Router Selector** matches only AxoRouter instances that have AxoStore enabled.

For details on creating a flow, see Manage flows.

Access the data

To access the data stored in AxoStore, see Storage. The data is stored according to the following schema: Storage schema.

Troubleshooting

If the data doesn't arrive at the destination, see the data doesn't reach the destination. To find the errors that the destination returns, see the destination rejects the data. For other problems, see Troubleshooting Axoflow.

8.4 ClickHouse

Find out about upcoming support for sending data from Axoflow to ClickHouse.

If you'd like to send data from AxoRouter to this destination, contact our support team for details.

8.5 CrowdStrike

Add a CrowdStrike Falcon Next-Gen SIEM destination, and look up the message fields it uses.

CrowdStrike: CrowdStrike Falcon Next-Gen SIEM unifies detection, investigation, and response on the Falcon platform. This destination sends logs to its HTTP Event Data Connector.

> **Note**
>
> **Axoflow can use CrowdStrike Falcon as a source and as a destination. To configure a CrowdStrike Falcon source, see CrowdStrike Falcon Platform.**

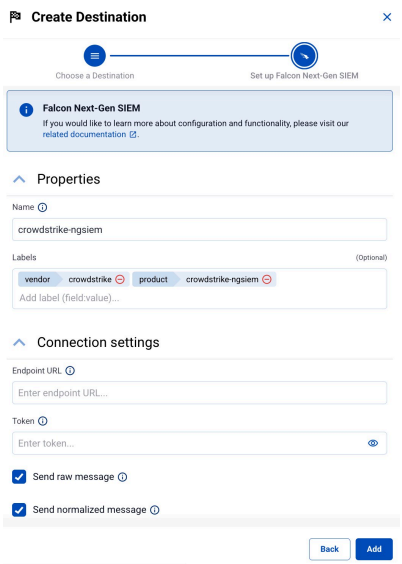
To add a CrowdStrike Falcon Next-Gen SIEM destination to Axoflow, complete the following steps. This destination sends data to the HTTP Event Data Connector (HEC) of your Falcon Next-Gen SIEM.

Prerequisites

1. Create a HTTP Event Data Connector (HEC) on your Falcon Next-Gen SIEM deployment if needed.
2. Record the API key and the ingest URL. You'll need them to configure Axoflow.

Steps

1. Create a new destination.
    1. Open the AxoConsole.
    2. Select **Destinations > Add Destination > Destination**.
2. Configure the destination.
    1. Select **Falcon Next-Gen SIEM**.
    2. Enter a name for the destination.



3. Enter your ingest URL into the **Endpoint URL** field. Enter the complete URL, because different regional URLs can follow different patterns.

4. Enter the API key you've created into the **Token** field.

5. By default, AxoRouter sends the logs:

   - In its original text form in the `event` element of the payload (**Send raw message** is enabled).
   - The ECS-normalized log in the `fields` element of the payload (**Send normalized message** is enabled).

6. (Optional) Set other options as needed for your environments.

   - **Disk buffer**: Explicitly enable or disable disk buffer on AxoRouter. Default value: **Enabled**.

     AxoRouter can store messages on the local disk if the destination or the network connection to the destination becomes unavailable. AxoRouter automatically sends the stored messages (in the order they were received) to the destination when the connection is reestablished. Note that the disk buffer is separate from AxoStore.

     Disk buffer capacity (per worker): Size of the disk buffer files in bytes. This value applies for each worker. Default value is 100MiB (104857600 bytes), which is also the recommended minimum.

     The disk required for using disk buffering is **Disk buffer capacity (per worker)** * **Number of Workers** for each destination of AxoRouter.

   - **Batch Size**: Sets the maximum size of payload in a batch. If the size of the messages reaches this value, AxoRouter sends the batch to the destination even if the number of messages is less than the value of the batch lines

option.

> **CAUTION:**
>
> When setting **Batch Size**:
>
> - Note that the actual HTTP request will be about 4KB larger because of the request headers.
> - Check the configuration of your destination, as SaaS solutions and ingress controllers often limit the maximal size of HTTP requests. If the sum of **Batch Size** and the request headers exceeds this limit, the data will be rejected. (In case of NGINX, with a `client intended to send too large body` error message.)

- **Batch Lines**: Number of lines sent to the destination in one batch. AxoRouter waits for this number of lines to accumulate and sends them off in a single batch. Increasing this number increases throughput as more messages are sent in a single batch, but also increases message latency.

  Note that **Initial window size** * **Maximum Connections** must be greater or equal than **Batch lines** * **Number of workers** for all the sources that feeds a given destination. Otherwise, the destination worker might be unable to send data until **Batch timeout** expires, even if there's data available.

  For more details, see the AxoSyslog documentation.

- **Batch Timeout**: Maximal time in milliseconds to wait for a batch to be filled before sending it. The default value (`-1`) means that the batch should be full before sending, which may result in a long wait if the incoming traffic is low.

  For more details, see the AxoSyslog documentation.

- **Content compression**: Specifies the compression to use when sending data to the destination. `No compression` explicitly disables compression. Other possible values: `Gzip` , `Deflate` . Select only compression that your destination supports and has enabled. Default value: `No compression`

- **Number of retries**: The number of retry attempts when the retry action is triggered. Default: `3` .

  For more details, see the AxoSyslog documentation.

- **Time to reopen**: The time to wait before reconnecting when a connection fails. Default: `1` second.

  For more details, see the AxoSyslog documentation.

- **Timeout**: The number of seconds to wait for a log-forwarding request to complete, and attempt to reconnect the server if exceeded. If the timeout is exceeded, AxoRouter attempts to reconnect the destination. The default (`0`) is unlimited.

  For more details, see the AxoSyslog documentation.

- **Number of Workers**: Used for scaling the destination in case of high message load. Specifies the number of worker threads AxoRouter uses for sending messages to the destination. The default is 1. If high message throughput is not a concern, leave it on 1. For maximum performance, increase it up to the number of CPU cores available on AxoRouter.

  For more details, see the AxoSyslog documentation.

- **Response actions**: Specifies what AxoRouter does with the log message, based on the response code received from the destination.

  - If the status code begins with 2 (for example, `200`), AxoRouter assumes the message was successfully sent.
  - If you've configured a custom action in AxoConsole for the status code, AxoRouter performs the specified action.
  - If the status code appears in the default response actions table, AxoRouter performs the specified action.
  - For other status codes, AxoRouter disconnects.

  To add a custom response action, select **Add new action**, then enter the **HTTP status code** and the related **Action**. The following actions are available:

  - **Disconnect**: Keep trying to resend the message indefinitely.
  - **Drop**: Drop the message without trying to resend it.
  - **Retry**: Retry sending the message for a maximum of 3 times.
  - **Success**: Assume the message was successfully sent.

> **Show default actions**
>
> | code | explanation | action |
> |---|---|---|
> | 100 | "Continue" | disconnect |
> | 101 | "Switching Protocols" | disconnect |
> | 102 | "Processing" | retry |
> | 103 | "Early Hints" | retry |
> | 200 | "OK" | success |

201	“Created”	success
202	“Accepted”	success
203	“Non-Authoritative Information”	success
204	“No Content”	success
205	“Reset Content”	success
206	“Partial Content”	success
300	“Multiple Choices”	disconnect
301	“Moved Permanently”	disconnect
302	“Found”	disconnect
303	“See Other”	disconnect
304	“Not Modified”	retry
307	“Temporary Redirect”	disconnect
308	“Permanent Redirect”	disconnect
400	“Bad Request”	disconnect
401	“Unauthorized”	disconnect
402	“Payment Required”	disconnect
403	“Forbidden”	disconnect
404	“Not Found”	disconnect
405	“Method Not Allowed”	disconnect
406	“Not Acceptable”	disconnect
407	“Proxy Authentication Required”	disconnect
408	“Request Timeout”	disconnect
409	“Conflict”	disconnect
410	“Gone”	drop
411	“Length Required”	disconnect

412	“Precondition Failed”	disconnect
413	“Payload Too Large”	disconnect
414	“URI Too Long”	disconnect
415	“Unsupported Media Type”	disconnect
416	“Range Not Satisfiable”	drop
417	“Expectation Failed”	disconnect
418	“I’m a teapot”	disconnect
421	“Misdirected Request”	disconnect
422	“Unprocessable Entity”	drop
423	“Locked”	disconnect
424	“Failed Dependency”	drop
425	“Too Early”	drop
426	“Upgrade Required”	disconnect
428	“Precondition Required”	retry
429	“Too Many Requests”	disconnect
431	“Request Header Fields Too Large”	disconnect
451	“Unavailable For Legal Reasons”	drop
500	“Internal Server Error”	disconnect
501	“Not Implemented”	disconnect
502	“Bad Gateway”	disconnect
503	“Service Unavailable”	disconnect
504	“Gateway Timeout”	retry
505	“HTTP Version Not Supported”	disconnect
506	“Variant Also Negotiates”	disconnect
507	“Insufficient Storage”	disconnect

508	“Loop Detected”	drop
510	“Not Extended”	disconnect
511	“Network Authentication Required”	disconnect

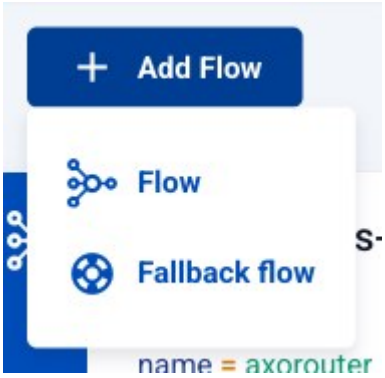
7. Select **Add**.

3. **Create a flow to connect the new destination to an AxoRouter instance.**

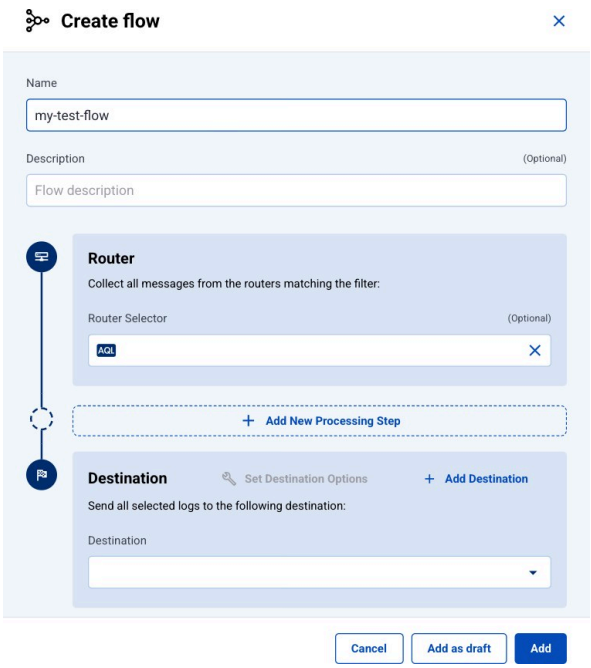
   1. Select **Flows**.

   2. Select **Add Flow > Flow**.

      To create a fallback flow, select **Add Flow > Fallback flow**.



   3. Enter a name for the flow, for example, `my-test-flow`.



   4. In the **Router Selector** field, enter an expression that matches the routers you want to apply the flow to. To select a specific router, use a name selector, for example, `name = my-axorouter-hostname`.

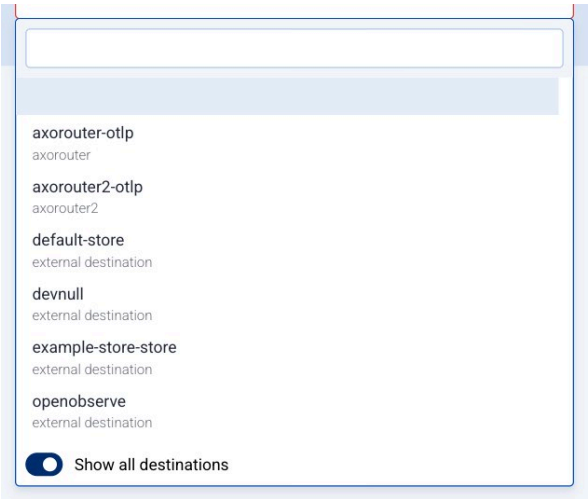
You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

- If you leave the router selector empty, it matches every AxoRouter instance.
- To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.
- If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

> **Note**
>
> **You can configure multiple fallback flows, but only one fallback flow can apply to an AxoRouter (so the Router Selector of the fallback flows can't overlap).**

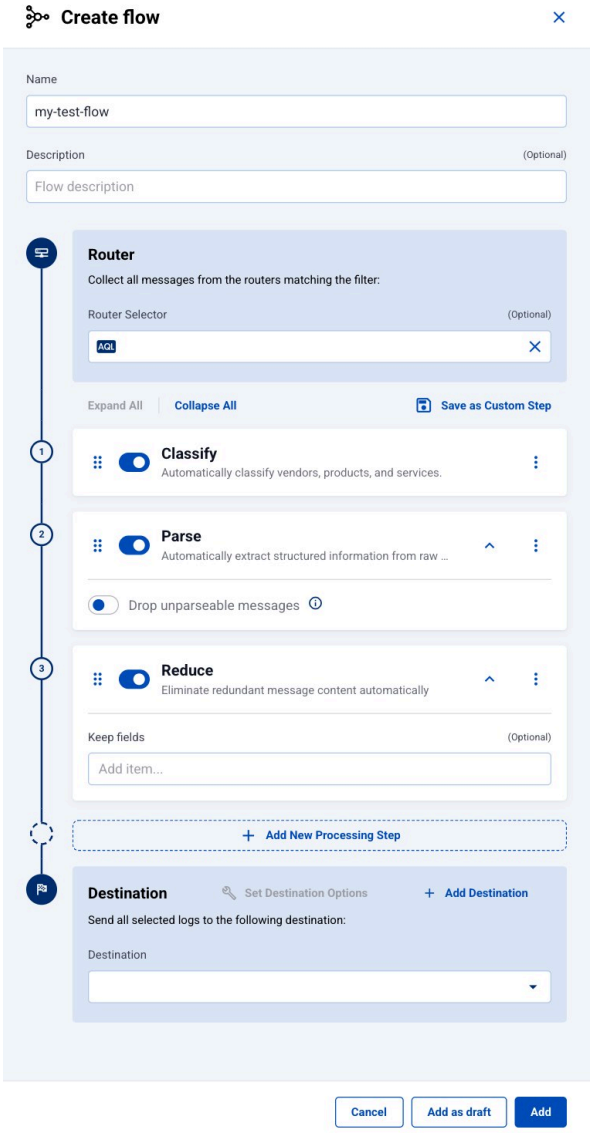
5. Select the **Destination** where you want to send your data. If you don't have any destination configured, you can select **Add Destination** in the destination section to create a new destination now. For details on the different destinations, see Destinations.

   - If you don't have any destination configured, see Destinations.
   - If you've already created a store, it's automatically available as a destination. Note that the **Router Selector** of the flow must match only AxoRouters that have the selected store available, otherwise you'll get an error message.
   - If you want to send data to another AxoRouter, enable the **Show all destinations** option, and select the connector of the AxoRouter where you want to send the data.



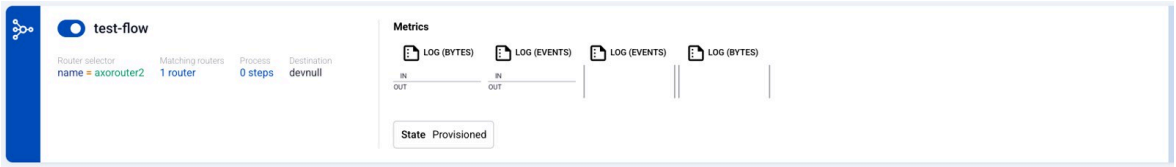
6. (Optional) To process the data transferred in the flow, select **Add New Processing Step**. For details, see Processing steps. For example:

    1. Add a **Classify**, a **Parse**, and a **Reduce** step, in that order, to automatically remove redundant and empty fields from your data.
    2. To select which messages are processed by the flow, add a **Select Messages** step, and enter a filter into the **AQL Expression** field. For example, to select only the messages received from Fortinet FortiGate firewalls, use the `meta.vendor = fortinet AND meta.product = fortigate` query.



7. Select **Add**, then confirm with **Add**. To save the flow without provisioning it to the routers yet, select **Add as draft** instead.

8. The new flow appears in the **Flows** list.



Related message fields

CrowdStrike HEC is a Splunk-compatible ingestion endpoint, so you can use the Splunk-related message fields to modify messages sent to this destination using processing steps.

Troubleshooting

If the data doesn't arrive at the destination, see the data doesn't reach the destination. To find the errors that the destination returns, see the destination rejects the data. For other problems, see Troubleshooting Axoflow.

8.6 Dynatrace

Add a Dynatrace destination to send your security data to Dynatrace, and look up the message fields it uses.

Dynatrace: Unified Observability — Drive business transformation with contextual analytics, AI, and automation.

To add a Dynatrace destination to Axoflow, complete the following steps.

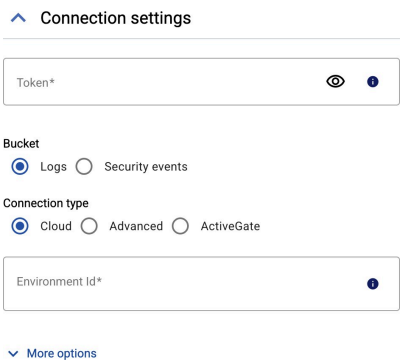
Prerequisites

- An access token to your Dynatrace instance. The token must have access to the Log ingestion API endpoint.
- Your AxoRouters that are sending data to Dynatrace must have network access to the following Log Ingestion endpoints:
  - For the **Cloud** collector, when you're sending data directly to the Dynatrace SaaS: `https://{your-environment-id}.live.dynatrace.com/api/v2/logs/ingest`
  - For the `ActiveGate` collector: `https://{your-activegate-domain}:9999/e/{your-environment-id}/api/v2/logs/ingest`

  If nothing prevents it, we recommend using the **Cloud** collector.
- Make sure to configure the limits of your Dynatrace instance to make sure it can ingest the amount of data you're sending.

Steps

1. Create a new destination.
   1. Open the AxoConsole.
   2. Select **Destinations > Add Destination > Destination**.
2. Configure the destination.
   1. Select **Dynatrace**.
   2. Enter a name for the destination.



3. Enter the token you've created into the **Token** field.

4. Select the Dynatrace **Bucket** type (**Logs** or **Security events**) where you want to send the data. Where exactly the data will be stored in Dynatrace depends on your Bucket assignment rules.

5. Select the type of Dynatrace **Collector** where you want to send the data:

   - **Cloud**: Send data directly to your Dynatrace SaaS subscription.
   - **ActiveGate**: Send data directly to a local ActiveGate endpoint.
   - **Advanced**: Specify a custom URL endpoint.

   If nothing prevents it, we recommend using the **Cloud** collector.

6. Configure the endpoint of the destination, depending on the collector type you've selected.

   - **Cloud**: Provide your **Environment Id** for your Dynatrace SaaS subscription.
   - **ActiveGate**:
     - Provide your **Environment Id** for your Dynatrace SaaS subscription.
     - Enter the **Domain** (or hostname) where your Dynatrace ActiveGate is running.
     - Enter the **Port** number where ActiveGate is listening on. Usually it's `9999`. and optionally the **Port** of the ActiveGate endpoint where you want to send your data.
   - **Advanced**: Specify a custom **URL** endpoint.

7. (Optional) Set other options as needed for your environments.

   - **Timeout**: The number of seconds to wait for a log-forwarding request to complete, and attempt to reconnect the server if exceeded. If the timeout is exceeded, AxoRouter attempts to reconnect the destination. The default (`0`) is unlimited.

     For more details, see the AxoSyslog documentation.

- **Disk buffer**: Explicitly enable or disable disk buffer on AxoRouter. Default value: **Enabled**.

  AxoRouter can store messages on the local disk if the destination or the network connection to the destination becomes unavailable. AxoRouter automatically sends the stored messages (in the order they were received) to the destination when the connection is reestablished. Note that the disk buffer is separate from AxoStore.

  Disk buffer capacity (per worker): Size of the disk buffer files in bytes. This value applies for each worker. Default value is 100MiB (104857600 bytes), which is also the recommended minimum.

  The disk required for using disk buffering is **Disk buffer capacity (per worker)** * **Number of Workers** for each destination of AxoRouter.

- **Batch Size**: Sets the maximum size of payload in a batch. If the size of the messages reaches this value, AxoRouter sends the batch to the destination even if the number of messages is less than the value of the batch lines option.

  > **CAUTION:**
  >
  > When setting **Batch Size**:
  >
  > - Note that the actual HTTP request will be about 4KB larger because of the request headers.
  > - Check the configuration of your destination, as SaaS solutions and ingress controllers often limit the maximal size of HTTP requests. If the sum of **Batch Size** and the request headers exceeds this limit, the data will be rejected. (In case of NGINX, with a `client intended to send too large body` error message.)

- **Batch Lines**: Number of lines sent to the destination in one batch. AxoRouter waits for this number of lines to accumulate and sends them off in a single batch. Increasing this number increases throughput as more messages are sent in a single batch, but also increases message latency.

  Note that **Initial window size** * **Maximum Connections** must be greater or equal than **Batch lines** * **Number of workers** for all the sources that feeds a given destination. Otherwise, the destination worker might be unable to send data until **Batch timeout** expires, even if there's data available.

  For more details, see the AxoSyslog documentation.

- **Batch Timeout**: Maximal time in milliseconds to wait for a batch to be filled before sending it. The default value (`-1`) means that the batch should be full before sending, which may result in a long wait if the incoming traffic is low.

  For more details, see the AxoSyslog documentation.

- **Content compression**: Specifies the compression to use when sending data to the destination. `No compression` explicitly disables compression. Other possible values: `Gzip` , `Deflate` . Select only compression that your destination supports and has enabled. Default value: `No compression`

- **Number of retries**: The number of retry attempts when the retry action is triggered. Default: `3` .

  For more details, see the AxoSyslog documentation.

- **Time to reopen**: The time to wait before reconnecting when a connection fails. Default: `1` second.

  For more details, see the AxoSyslog documentation.

- **Timeout**: The number of seconds to wait for a log-forwarding request to complete, and attempt to reconnect the server if exceeded. If the timeout is exceeded, AxoRouter attempts to reconnect the destination. The default (`0`) is unlimited.

  For more details, see the AxoSyslog documentation.

- **Number of Workers**: Used for scaling the destination in case of high message load. Specifies the number of worker threads AxoRouter uses for sending messages to the destination. The default is 1. If high message throughput is not a concern, leave it on 1. For maximum performance, increase it up to the number of CPU cores available on AxoRouter.

  For more details, see the AxoSyslog documentation.

- **Response actions**: Specifies what AxoRouter does with the log message, based on the response code received from the destination.

  - If the status code begins with 2 (for example, `200`), AxoRouter assumes the message was successfully sent.
  - If you've configured a custom action in AxoConsole for the status code, AxoRouter performs the specified action.
  - If the status code appears in the default response actions table, AxoRouter performs the specified action.
  - For other status codes, AxoRouter disconnects.

To add a custom response action, select **Add new action**, then enter the **HTTP status code** and the related **Action**. The following actions are available:

- **Disconnect**: Keep trying to resend the message indefinitely.
- **Drop**: Drop the message without trying to resend it.
- **Retry**: Retry sending the message for a maximum of 3 times.
- **Success**: Assume the message was successfully sent.

Show default actions

code	explanation	action
100	“Continue”	disconnect
101	“Switching Protocols”	disconnect
102	“Processing”	retry
103	“Early Hints”	retry
200	“OK”	success
201	“Created”	success
202	“Accepted”	success
203	“Non-Authoritative Information”	success
204	“No Content”	success
205	“Reset Content”	success
206	“Partial Content”	success
300	“Multiple Choices”	disconnect
301	“Moved Permanently”	disconnect
302	“Found”	disconnect
303	“See Other”	disconnect
304	“Not Modified”	retry
307	“Temporary Redirect”	disconnect
308	“Permanent Redirect”	disconnect
400	“Bad Request”	disconnect

401	“Unauthorized”	disconnect
402	“Payment Required”	disconnect
403	“Forbidden”	disconnect
404	“Not Found”	disconnect
405	“Method Not Allowed”	disconnect
406	“Not Acceptable”	disconnect
407	“Proxy Authentication Required”	disconnect
408	“Request Timeout”	disconnect
409	“Conflict”	disconnect
410	“Gone”	drop
411	“Length Required”	disconnect
412	“Precondition Failed”	disconnect
413	“Payload Too Large”	disconnect
414	“URI Too Long”	disconnect
415	“Unsupported Media Type”	disconnect
416	“Range Not Satisfiable”	drop
417	“Expectation Failed”	disconnect
418	“I’m a teapot”	disconnect
421	“Misdirected Request”	disconnect
422	“Unprocessable Entity”	drop
423	“Locked”	disconnect
424	“Failed Dependency”	drop
425	“Too Early”	drop
426	“Upgrade Required”	disconnect
428	“Precondition Required”	retry

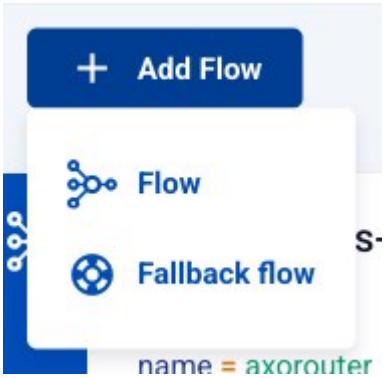
429	“Too Many Requests”	disconnect
431	“Request Header Fields Too Large”	disconnect
451	“Unavailable For Legal Reasons”	drop
500	“Internal Server Error”	disconnect
501	“Not Implemented”	disconnect
502	“Bad Gateway”	disconnect
503	“Service Unavailable”	disconnect
504	“Gateway Timeout”	retry
505	“HTTP Version Not Supported”	disconnect
506	“Variant Also Negotiates”	disconnect
507	“Insufficient Storage”	disconnect
508	“Loop Detected”	drop
510	“Not Extended”	disconnect
511	“Network Authentication Required”	disconnect

3. **Create a flow to connect the new destination to an AxoRouter instance.**

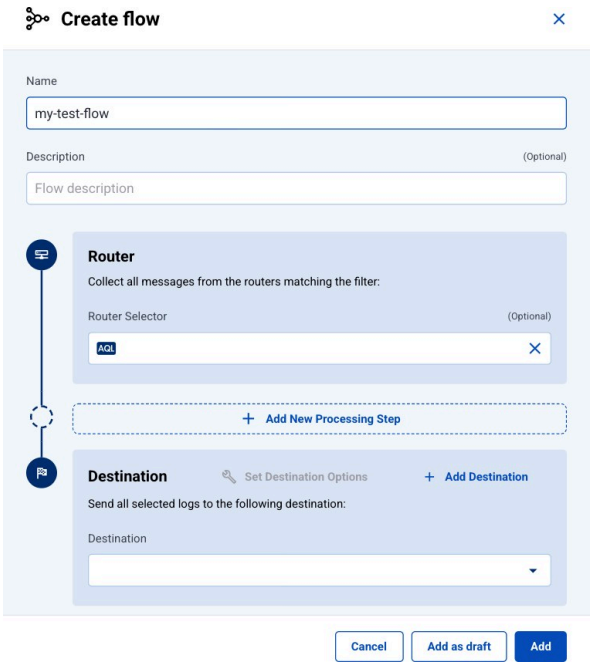
   1. Select **Flows**.

   2. Select **Add Flow > Flow**.

      To create a fallback flow, select **Add Flow > Fallback flow**.

   3. Enter a name for the flow, for example, `my-test-flow`.



4. In the **Router Selector** field, enter an expression that matches the routers you want to apply the flow to. To select a specific router, use a name selector, for example, `name = my-axorouter-hostname`.

   You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

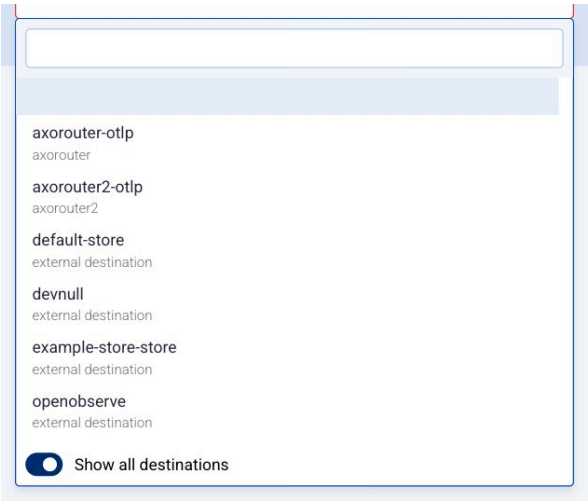
   - If you leave the router selector empty, it matches every AxoRouter instance.
   - To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.
   - If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

   > **Note**
   >
   > **You can configure multiple fallback flows, but only one fallback flow can apply to an AxoRouter (so the Router Selector of the fallback flows can't overlap).**

5. Select the **Destination** where you want to send your data. If you don't have any destination configured, you can select **Add Destination** in the destination section to create a new destination now. For details on the different destinations, see Destinations.

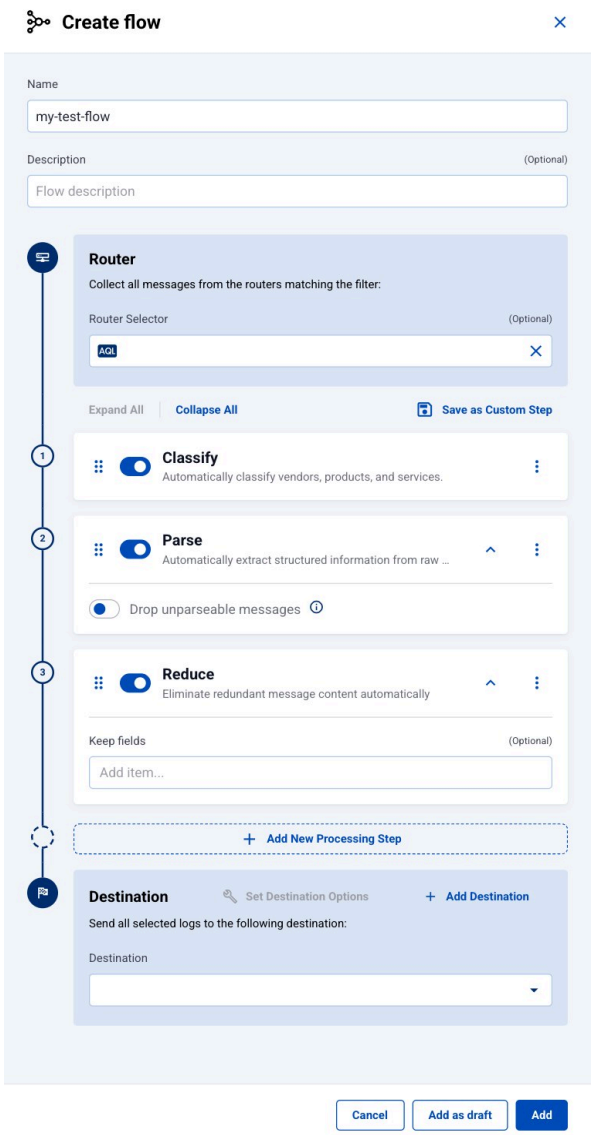
   - If you don't have any destination configured, see Destinations.

- If you've already created a store, it's automatically available as a destination. Note that the **Router Selector** of the flow must match only AxoRouters that have the selected store available, otherwise you'll get an error message.
- If you want to send data to another AxoRouter, enable the **Show all destinations** option, and select the connector of the AxoRouter where you want to send the data.



6. (Optional) To process the data transferred in the flow, select **Add New Processing Step**. For details, see Processing steps. For example:

   1. Add a **Classify**, a **Parse**, and a **Reduce** step, in that order, to automatically remove redundant and empty fields from your data.
   2. To select which messages are processed by the flow, add a **Select Messages** step, and enter a filter into the **AQL Expression** field. For example, to select only the messages received from Fortinet FortiGate firewalls, use the `meta.vendor = fortinet AND meta.product = fortigate` query.



7. Select **Add**, then confirm with **Add**. To save the flow without provisioning it to the routers yet, select **Add as draft** instead.

8. The new flow appears in the **Flows** list.



Related message fields

You can use the following message fields to modify messages sent to this destination using processing steps.

Troubleshooting

If the data doesn't arrive at the destination, see the data doesn't reach the destination. To find the errors that the destination returns, see the destination rejects the data. For other problems, see Troubleshooting Axoflow.

8.7 Elastic

Send data from Axoflow to Elastic Cloud and self-managed Elasticsearch clusters.

8.7.1 Elastic Cloud and Elasticsearch

Add an Elasticsearch destination for Elastic Cloud or a self-managed cluster, and look up the message fields it uses.

Elastic Cloud and Elasticsearch: Distributed search and analytics engine for real-time log analysis, observability, and data indexing.

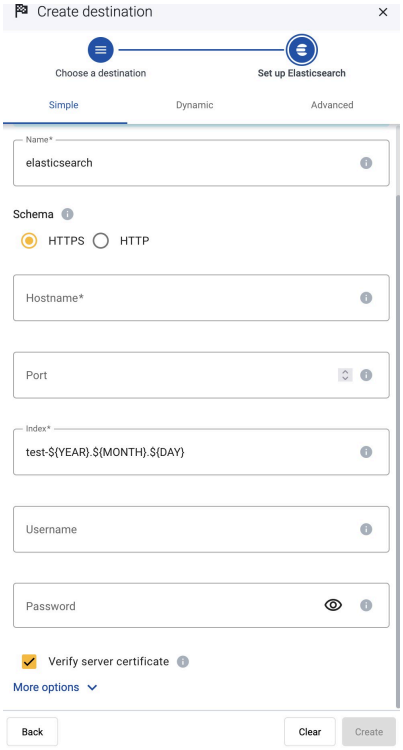
To add an Elasticsearch destination to Axoflow, complete the following steps.

Prerequisites

- An Elastic Cloud account for Axoflow, or
- a self-hosted Elasticsearch deployment.
- To configure Axoflow, you'll need the username, password, and the Elasticsearch index where you want to send your data.

Steps

1. Create a new destination.
    1. Open the AxoConsole.
    2. Select **Destinations > Add Destination > Destination**.
2. Configure the destination.
    1. Select **Elasticsearch**.
    2. Select which type of configuration you want to use:
        - **Simple**: Send all data into a single index.
        - **Dynamic**: Send data to an index based on the content or metadata of the incoming messages (or to a default index).
        - **Advanced**: Allows you to specify a custom URL endpoint.
    3. Enter a name for the destination.



4. Configure the endpoint of the destination.

   - **Advanced**: Enter your Elasticsearch URL into the **URL** field, for example, `http://my-elastic-server:9200/_bulk`
   - **Simple** and **Dynamic**:
     1. Select the **HTTPS** or **HTTP** protocol to use to access your destination.
     2. Enter the **Hostname** and **Port** of the destination.

5. Specify the Elasticsearch index to send the data to.

   - **Simple**: Enter the expression that specifies the Elasticsearch index to use into the **Index** field, for example: `test-${YEAR}${MONTH}${DAY}`. All data will be sent into this index.
   - **Dynamic** and **Advanced**: Enter the expression that specifies the default index. The data will be sent into this index if no other index is set during the processing of the message (for example, by the processing steps of the **Flow**).

   You can use AxoSyslog macros in this field.

6. Enter the username and password for the account you want to use.

7. (Optional)

   By default, Axoflow rejects connections to the destination server if the certificate of the server is invalid (for example, it's expired, signed by an unknown CA, or its CN and the name of the server is mismatched: the `Common Name` or the `subject_alt_name` parameter of the peer's certificate must contain the hostname or the IP address (as resolved from AxoRouter) of the peer).

> **Warning**
>
> **When validating a certificate, the entire certificate chain must be valid, including the CA certificate. If any certificate of the chain is invalid, Axoflow will reject the connection.**

If you want to accept invalid certificates (or no certificate) from the destination servers, disable the **Verify server certificate** option.

8. (Optional) Set other options as needed for your environments.

   - **Timeout**: The number of seconds to wait for a log-forwarding request to complete, and attempt to reconnect the server if exceeded. If the timeout is exceeded, AxoRouter attempts to reconnect the destination. The default (`0`) is unlimited.

     For more details, see the AxoSyslog documentation.

   - **Disk buffer**: Explicitly enable or disable disk buffer on AxoRouter. Default value: **Enabled**.

     AxoRouter can store messages on the local disk if the destination or the network connection to the destination becomes unavailable. AxoRouter automatically sends the stored messages (in the order they were received) to the destination when the connection is reestablished. Note that the disk buffer is separate from AxoStore.

     Disk buffer capacity (per worker): Size of the disk buffer files in bytes. This value applies for each worker. Default value is 100MiB (104857600 bytes), which is also the recommended minimum.

     The disk required for using disk buffering is **Disk buffer capacity (per worker)** * **Number of Workers** for each destination of AxoRouter.

   - **Batch Size**: Sets the maximum size of payload in a batch. If the size of the messages reaches this value, AxoRouter sends the batch to the destination even if the number of messages is less than the value of the batch lines option.

     > **CAUTION:**
     >
     > When setting **Batch Size**:
     >
     > - Note that the actual HTTP request will be about 4KB larger because of the request headers.

- Check the configuration of your destination, as SaaS solutions and ingress controllers often limit the maximal size of HTTP requests. If the sum of **Batch Size** and the request headers exceeds this limit, the data will be rejected. (In case of NGINX, with a `client intended to send too large body` error message.)

- **Batch Lines**: Number of lines sent to the destination in one batch. AxoRouter waits for this number of lines to accumulate and sends them off in a single batch. Increasing this number increases throughput as more messages are sent in a single batch, but also increases message latency.

  Note that **Initial window size** * **Maximum Connections** must be greater or equal than **Batch lines** * **Number of workers** for all the sources that feeds a given destination. Otherwise, the destination worker might be unable to send data until **Batch timeout** expires, even if there's data available.

  For more details, see the AxoSyslog documentation.

- **Batch Timeout**: Maximal time in milliseconds to wait for a batch to be filled before sending it. The default value (`-1`) means that the batch should be full before sending, which may result in a long wait if the incoming traffic is low.

  For more details, see the AxoSyslog documentation.

- **Content compression**: Specifies the compression to use when sending data to the destination. `No compression` explicitly disables compression. Other possible values: `Gzip` , `Deflate` . Select only compression that your destination supports and has enabled. Default value: `No compression`

- **Number of retries**: The number of retry attempts when the retry action is triggered. Default: `3` .

  For more details, see the AxoSyslog documentation.

- **Time to reopen**: The time to wait before reconnecting when a connection fails. Default: `1` second.

  For more details, see the AxoSyslog documentation.

- **Timeout**: The number of seconds to wait for a log-forwarding request to complete, and attempt to reconnect the server if exceeded. If the timeout is exceeded, AxoRouter attempts to reconnect the destination. The default (`0`) is unlimited.

  For more details, see the AxoSyslog documentation.

- **Number of Workers**: Used for scaling the destination in case of high message load. Specifies the number of worker threads AxoRouter uses for sending messages to the destination. The default is 1. If high message throughput is not a concern, leave it on 1. For maximum performance, increase it up to the number of CPU cores available on AxoRouter.

  For more details, see the AxoSyslog documentation.

- **Response actions**: Specifies what AxoRouter does with the log message, based on the response code received from the destination.

  - If the status code begins with 2 (for example, `200`), AxoRouter assumes the message was successfully sent.
  - If you've configured a custom action in AxoConsole for the status code, AxoRouter performs the specified action.
  - If the status code appears in the default response actions table, AxoRouter performs the specified action.
  - For other status codes, AxoRouter disconnects.

  To add a custom response action, select **Add new action**, then enter the **HTTP status code** and the related **Action**. The following actions are available:

  - **Disconnect**: Keep trying to resend the message indefinitely.
  - **Drop**: Drop the message without trying to resend it.
  - **Retry**: Retry sending the message for a maximum of 3 times.
  - **Success**: Assume the message was successfully sent.

> **Show default actions**
>
> | code | explanation | action |
> |---|---|---|
> | 100 | "Continue" | disconnect |
> | 101 | "Switching Protocols" | disconnect |
> | 102 | "Processing" | retry |
> | 103 | "Early Hints" | retry |
> | 200 | "OK" | success |
> | 201 | "Created" | success |
> | 202 | "Accepted" | success |
> | 203 | "Non-Authoritative Information" | success |

204	“No Content”	success
205	“Reset Content”	success
206	“Partial Content”	success
300	“Multiple Choices”	disconnect
301	“Moved Permanently”	disconnect
302	“Found”	disconnect
303	“See Other”	disconnect
304	“Not Modified”	retry
307	“Temporary Redirect”	disconnect
308	“Permanent Redirect”	disconnect
400	“Bad Request”	disconnect
401	“Unauthorized”	disconnect
402	“Payment Required”	disconnect
403	“Forbidden”	disconnect
404	“Not Found”	disconnect
405	“Method Not Allowed”	disconnect
406	“Not Acceptable”	disconnect
407	“Proxy Authentication Required”	disconnect
408	“Request Timeout”	disconnect
409	“Conflict”	disconnect
410	“Gone”	drop
411	“Length Required”	disconnect
412	“Precondition Failed”	disconnect
413	“Payload Too Large”	disconnect
414	“URI Too Long”	disconnect

415	“Unsupported Media Type”	disconnect
416	“Range Not Satisfiable”	drop
417	“Expectation Failed”	disconnect
418	“I’m a teapot”	disconnect
421	“Misdirected Request”	disconnect
422	“Unprocessable Entity”	drop
423	“Locked”	disconnect
424	“Failed Dependency”	drop
425	“Too Early”	drop
426	“Upgrade Required”	disconnect
428	“Precondition Required”	retry
429	“Too Many Requests”	disconnect
431	“Request Header Fields Too Large”	disconnect
451	“Unavailable For Legal Reasons”	drop
500	“Internal Server Error”	disconnect
501	“Not Implemented”	disconnect
502	“Bad Gateway”	disconnect
503	“Service Unavailable”	disconnect
504	“Gateway Timeout”	retry
505	“HTTP Version Not Supported”	disconnect
506	“Variant Also Negotiates”	disconnect
507	“Insufficient Storage”	disconnect
508	“Loop Detected”	drop
510	“Not Extended”	disconnect
511	“Network Authentication Required”	disconnect

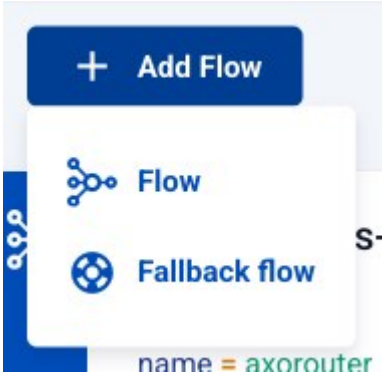
- **Message format**: Specifies the message format and normalization. Possible values:
  - **Elastic Common Schema (ECS)**: Normalize the data using the ECS schema.
  - **Open Cybersecurity Schema Framework (OCSF)**: Normalize the data using the OCSF schema.
  - **Parsed**: Send the parsed data without further normalization.

9. Select **Add**.

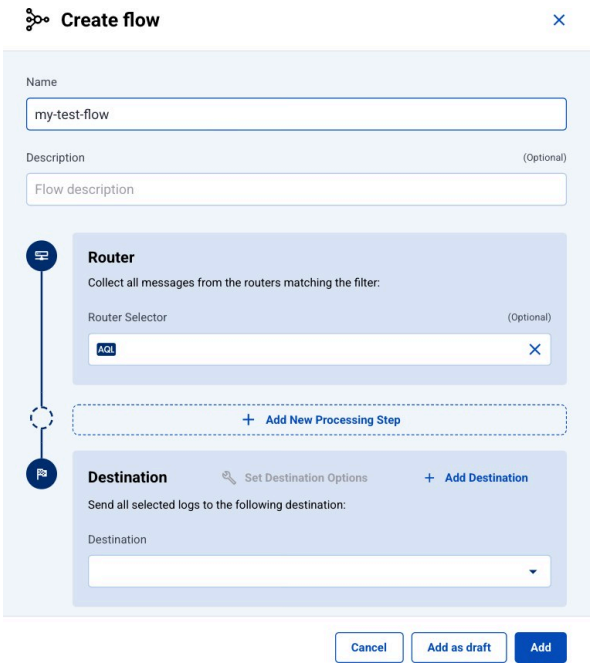
3. **Create a flow to connect the new destination to an AxoRouter instance.**

   1. Select **Flows**.
   2. Select **Add Flow > Flow**.

      To create a fallback flow, select **Add Flow > Fallback flow**.

   3. Enter a name for the flow, for example, `my-test-flow`.

4. In the **Router Selector** field, enter an expression that matches the routers you want to apply the flow to. To select a specific router, use a name selector, for example, `name = my-axorouter-hostname`.

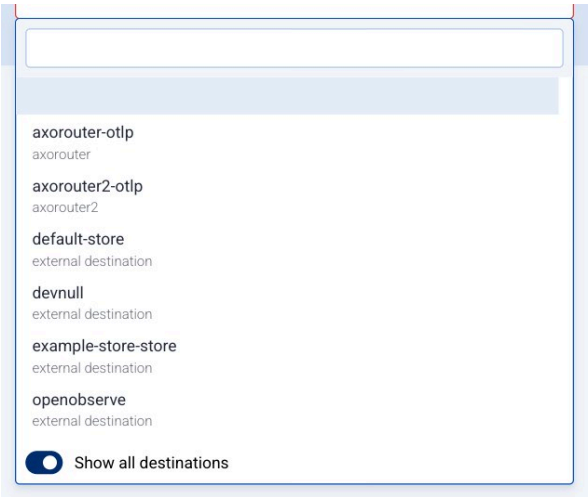
   You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

   - If you leave the router selector empty, it matches every AxoRouter instance.
   - To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.
   - If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

   > **Note**
   >
   > **You can configure multiple fallback flows, but only one fallback flow can apply to an AxoRouter (so the Router Selector of the fallback flows can't overlap).**

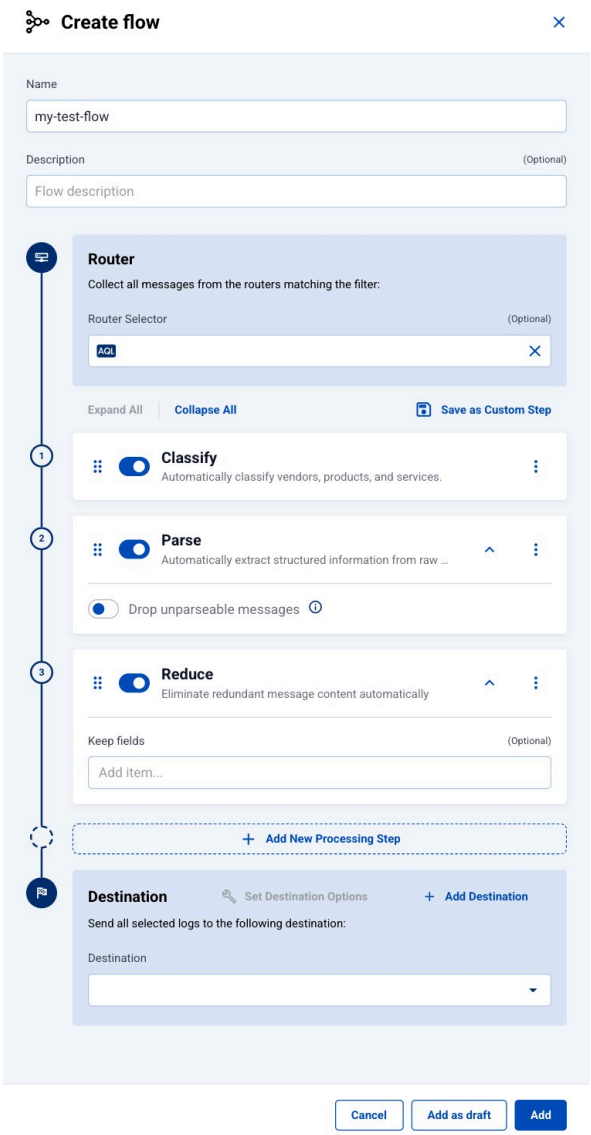
5. Select the **Destination** where you want to send your data. If you don't have any destination configured, you can select **Add Destination** in the destination section to create a new destination now. For details on the different destinations, see Destinations.

   - If you don't have any destination configured, see Destinations.
   - If you've already created a store, it's automatically available as a destination. Note that the **Router Selector** of the flow must match only AxoRouters that have the selected store available, otherwise you'll get an error message.
   - If you want to send data to another AxoRouter, enable the **Show all destinations** option, and select the connector of the AxoRouter where you want to send the data.



6. (Optional) To process the data transferred in the flow, select **Add New Processing Step**. For details, see Processing steps. For example:

    1. Add a **Classify**, a **Parse**, and a **Reduce** step, in that order, to automatically remove redundant and empty fields from your data.
    2. To select which messages are processed by the flow, add a **Select Messages** step, and enter a filter into the **AQL Expression** field. For example, to select only the messages received from Fortinet FortiGate firewalls, use the `meta.vendor = fortinet AND meta.product = fortigate` query.



7. Select **Add**, then confirm with **Add**. To save the flow without provisioning it to the routers yet, select **Add as draft** instead.

8. The new flow appears in the **Flows** list.



Related message fields

You can use the following message fields to modify messages sent to this destination using processing steps.

Troubleshooting

If the data doesn't arrive at the destination, see the data doesn't reach the destination. To find the errors that the destination returns, see the destination rejects the data. For other problems, see Troubleshooting Axoflow.

8.8 Generic

Forward data to any syslog receiver, or discard it with the /dev/null destination.

8.8.1 /dev/null

Discard data with a destination that accepts everything, for example, to test a flow or to find a bottleneck.

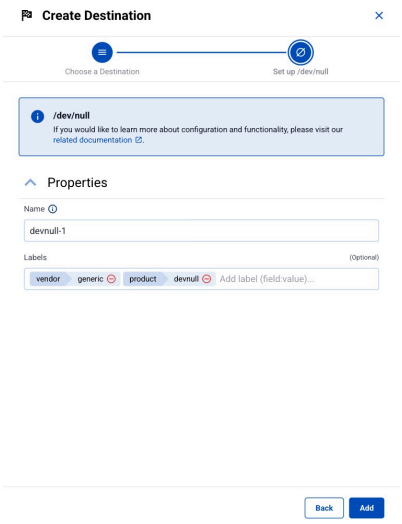
/dev/null: Blackhole destination for testing or discarding events without storing them.

This is a null destination that discards (drops) every data it receives, but reports that it has successfully received the data. This is useful sometimes for testing and performance measurements, for example, to find out if a real destination is the bottleneck, or another element in the upstream pipeline.

> **CAUTION:**
>
> All data that's sent to the /dev/null destination only is irrevocably lost.

Steps

1. Create a new destination.
    1. Open the AxoConsole.
    2. Select **Destinations > Add Destination > Destination**.
2. Configure the destination.
    1. Select **/dev/null**.
    2. Enter a name for the destination.



3. (Optional): Add custom labels to the destination.

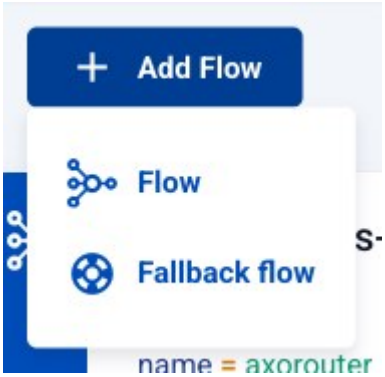
4. Select **Add**.

3. **Create a flow to connect the new destination to an AxoRouter instance.**

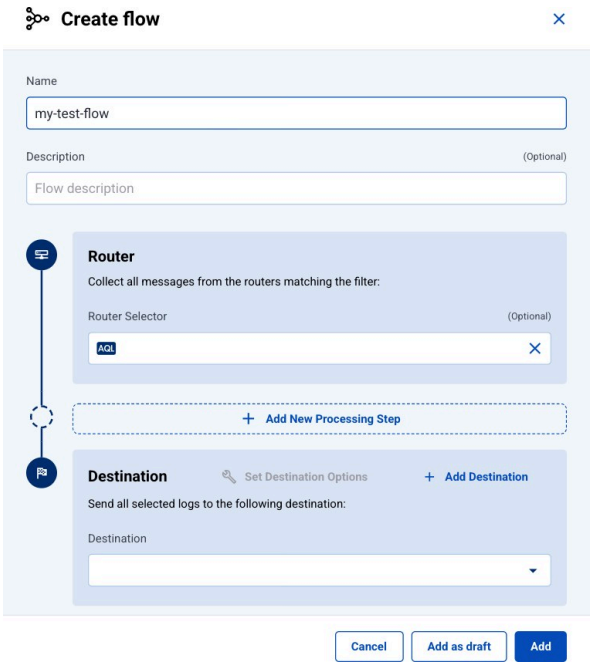
   1. Select **Flows**.

   2. Select **Add Flow > Flow**.

      To create a fallback flow, select **Add Flow > Fallback flow**.



   3. Enter a name for the flow, for example, `my-test-flow`.



   4. In the **Router Selector** field, enter an expression that matches the routers you want to apply the flow to. To select a specific router, use a name selector, for example, `name = my-axorouter-hostname`.

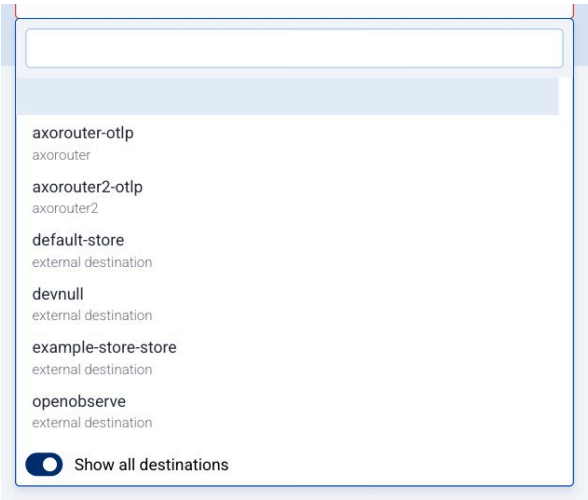
      You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

- If you leave the router selector empty, it matches every AxoRouter instance.
- To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.
- If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

> **Note**
>
> **You can configure multiple fallback flows, but only one fallback flow can apply to an AxoRouter (so the Router Selector of the fallback flows can't overlap).**

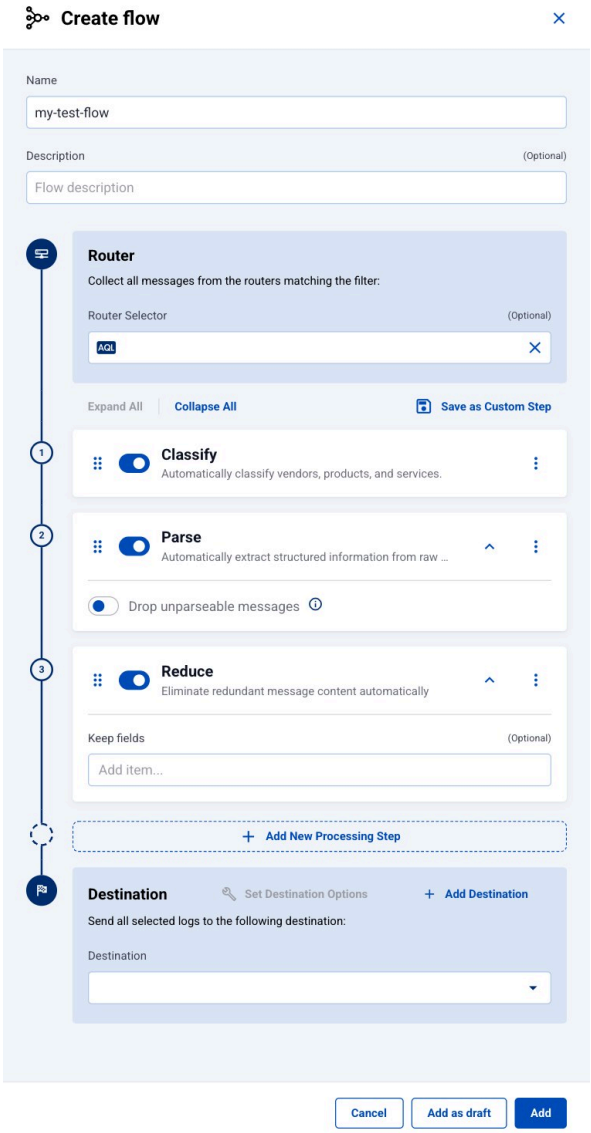
5. Select the **Destination** where you want to send your data. If you don't have any destination configured, you can select **Add Destination** in the destination section to create a new destination now. For details on the different destinations, see Destinations.

   - If you don't have any destination configured, see Destinations.
   - If you've already created a store, it's automatically available as a destination. Note that the **Router Selector** of the flow must match only AxoRouters that have the selected store available, otherwise you'll get an error message.
   - If you want to send data to another AxoRouter, enable the **Show all destinations** option, and select the connector of the AxoRouter where you want to send the data.

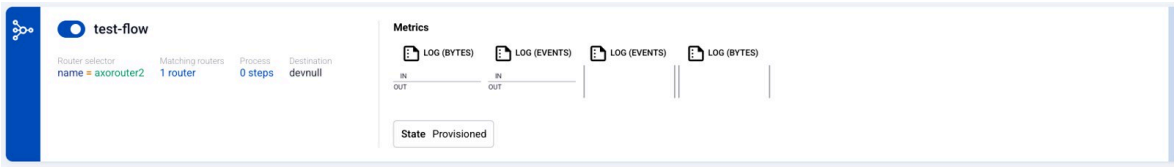


6. (Optional) To process the data transferred in the flow, select **Add New Processing Step**. For details, see Processing steps. For example:

1. Add a **Classify**, a **Parse**, and a **Reduce** step, in that order, to automatically remove redundant and empty fields from your data.
2. To select which messages are processed by the flow, add a **Select Messages** step, and enter a filter into the **AQL Expression** field. For example, to select only the messages received from Fortinet FortiGate firewalls, use the `meta.vendor = fortinet AND meta.product = fortigate` query.



7. Select **Add**, then confirm with **Add**. To save the flow without provisioning it to the routers yet, select **Add as draft** instead.
8. The new flow appears in the **Flows** list.



8.8.2 syslog

Forward data in RFC 3164 or RFC 5424 syslog format over UDP, TCP, or TLS, and look up the protocol-specific options.

syslog: Standard protocol for system logs. Our source automatically detects, parses, and classifies incoming syslog events without pre-configuration.

The syslog destination forwards your security data in an RFC-3164 or RFC-5424 compliant syslog format, using the UDP, TCP, or TLS-encrypted TCP protocols.

Prerequisites

If you want to enable TLS encryption for this connector to encrypt the communication with the sources, you'll need to set appropriate keys and certificates.

> **CAUTION:**
>
> Copy the keys and certificates to AxoRouter before starting to configure the connector. Otherwise, you won't be able to make configuration changes that require reloading the AxoRouter service, including starting log tapping or flow tapping.

Note the following points:

- Keys and certificates must be in PEM format.
- If the file contains a certificate chain, the file must begin with the certificate of the host, followed by the CA certificate that signed the certificate of the host, and any other signing CAs in order.
- If you manually copy these files to their place on the AxoRouter host, make the files readable by the `axorouter` service.
  - The recommended path for certificates is under `/etc/axorouter/user-config/` (for example, `/etc/axorouter/user-config/tls-key.pem`). (If you need to use a different path, you have to append an option like `-v /your/path:/your/path` to the `AXOROUTER_PODMAN_ARGS` variable of `/etc/axorouter/container.env`.)
  - When referring to the key or certificate when configuring the connector, use absolute paths (for example, `/etc/axorouter/user-config/tls-key.pem`).

Steps

1. Create a new destination.
   1. Open the AxoConsole.

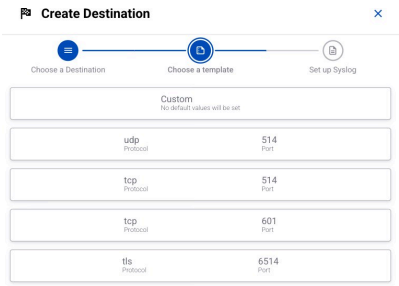
2. Select **Destinations > Add Destination > Destination**.

2. Configure the destination.

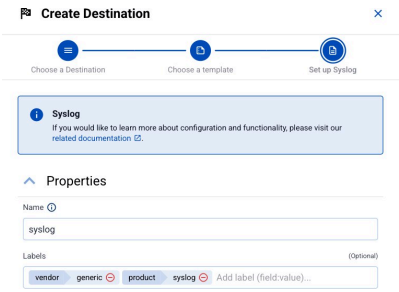
    1. Select **Syslog**.

    2. Select the template to use one of the standard syslog ports and transport protocols: UDP 514 (commonly used for the RFC3164 syslog protocol), TCP 514, TCP 601, or TLS 6514.

        To configure a different port, or to specify the protocol elements manually, select **Custom**.

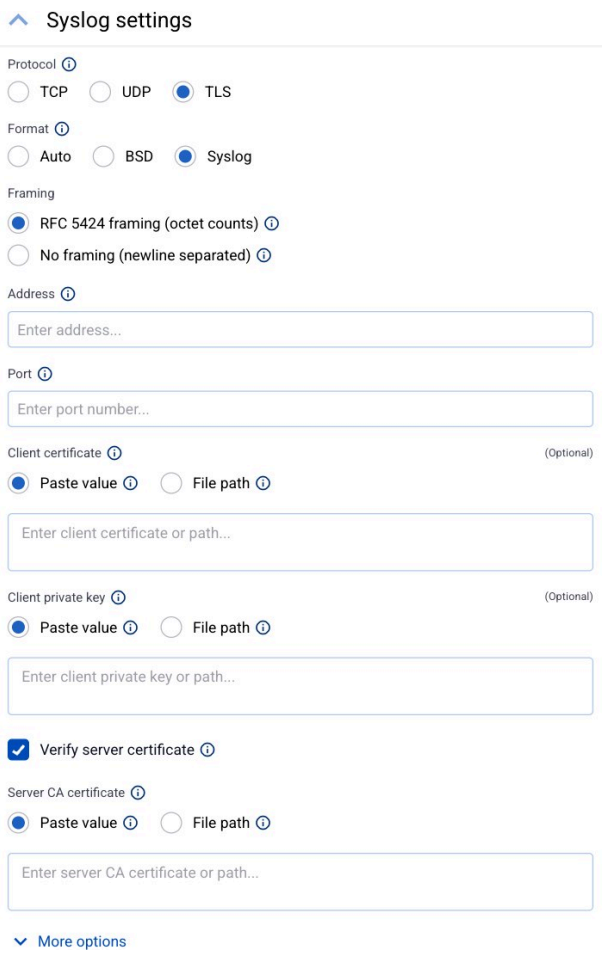


    3. Enter a name for the destination.



    4. (Optional): Add custom labels to the destination.

    5. Select the protocol to use for sending syslog data: **TCP**, **UDP**, or **TLS**.



6. Select the syslog format to use: **Auto**, **BSD** (RFC3164), or **Syslog** (RFC5424). The default **Auto** option sends messages in the **Syslog** (RFC5424) format.

7. (Optional) If explicitly needed for your use case, you can configure **Framing** manually when using the **Auto** or the **Syslog** (RFC5424) format over TCP or TLS. Select **RFC 5424 framing (octet counts)** if the payload contains the length of the message as specified in RFC6587 3.4.1. Select **No framing (newline separated)** for non-transparent-framing RFC6587 3.4.2.

8. If you've selected **Protocol > TLS**, set the TLS-related options.

   When using TLS, set the paths for the certificates and keys used for the TLS-encrypted communication with the clients. For details, see Prerequisites.

   - **Client certificate**: The certificate that AxoRouter shows to the destination server.

     - To provide the client certificate from the AxoConsole, select **Paste value** and paste the client certificate. AxoConsole will automatically deploy the client certificate to the related AxoRouters.
     - To provide the client certificate from a file, select **File path** and enter the path where the certificate file is located (for example, `/etc/axorouter/user-config/client-certificate.pem`). The path must be

mounted into the container, `/etc/axorouter` is mounted by default. You must manually copy that file to its place on every affected AxoRouter, or you can use the configuration management system of your choice.

- **Client private key**: The private key of the client certificate.
  - To provide the client's private key from the AxoConsole, select **Paste value** and paste the client's private key. AxoConsole will automatically deploy the client's private key to the related AxoRouters.
  - To provide the client's private key from a file, select **File path** and enter the path where the private key file is located (for example, `/etc/axorouter/user-config/client-private-key.pem`). The path must be mounted into the container, `/etc/axorouter` is mounted by default. You must manually copy that file to its place on every affected AxoRouter, or you can use the configuration management system of your choice.
- **Server CA certificate**: The CA certificate that AxoRouter uses to verify the certificate of the destination if **Verify server certificate** is enabled.
  - To provide the CA certificate from the AxoConsole, select **Paste value** and paste the CA certificate. AxoConsole will automatically deploy the CA certificate to the related AxoRouters.
  - To provide the CA certificate from a file, select **File path** and enter the path where the certificate file is located (for example, `/etc/axorouter/user-config/ca-certificate.pem`). The path must be mounted into the container, `/etc/axorouter` is mounted by default. You must manually copy that file to its place on every affected AxoRouter, or you can use the configuration management system of your choice.

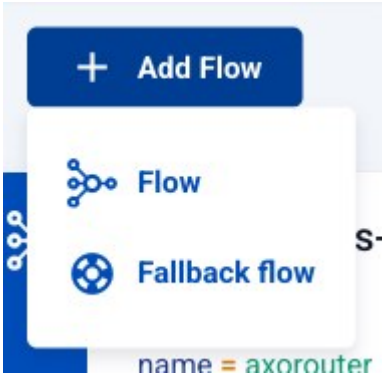
9. Set the **Address** and the **Port** of the destination. Usually:
   - 514 UDP and TCP for RFC3164 (BSD-syslog) and RFC5424 (IETF-syslog) formatted traffic. AxoRouter automatically recognizes and handles both formats.
   - 601 TCP (RFC 3195 - Reliable Delivery for syslog) for RFC5424 (IETF-syslog) and RFC3164 (BSD-syslog) formatted traffic. AxoRouter automatically recognizes and handles both formats.
   - 6514 TCP for TLS-encrypted syslog traffic (RFC 5425).
10. (Optional) To rate-limit the output of the destination, select **More options**, then set the number of messages (per second) to send in the **Throttle** field.
11. Select **Add**.

3. **Create a flow to connect the new destination to an AxoRouter instance.**

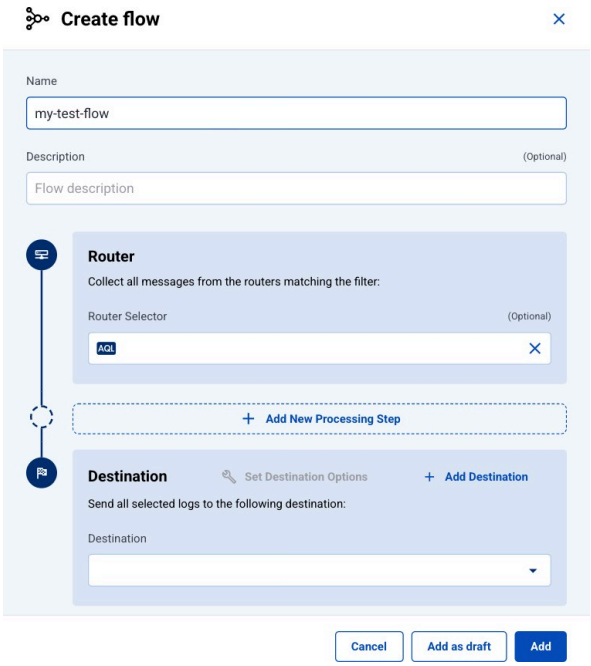
   1. Select **Flows**.

2. Select **Add Flow > Flow**.

   To create a fallback flow, select **Add Flow > Fallback flow**.



3. Enter a name for the flow, for example, `my-test-flow`.



4. In the **Router Selector** field, enter an expression that matches the routers you want to apply the flow to. To select a specific router, use a name selector, for example, `name = my-axorouter-hostname`.

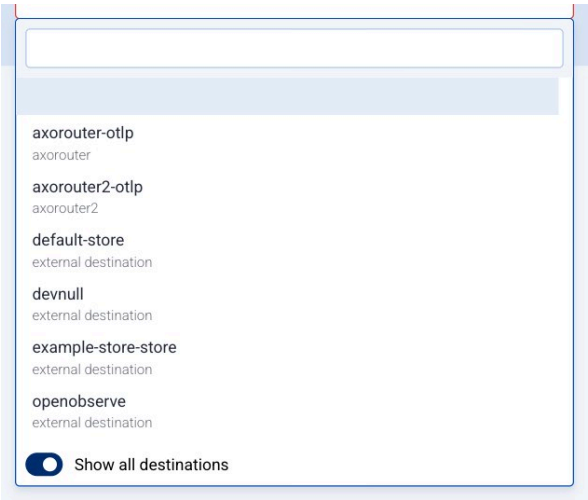
   You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

   - If you leave the router selector empty, it matches every AxoRouter instance.
   - To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.
   - If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

> **Note**
>
> **You can configure multiple fallback flows, but only one fallback flow can apply to an AxoRouter (so the Router Selector of the fallback flows can't overlap).**

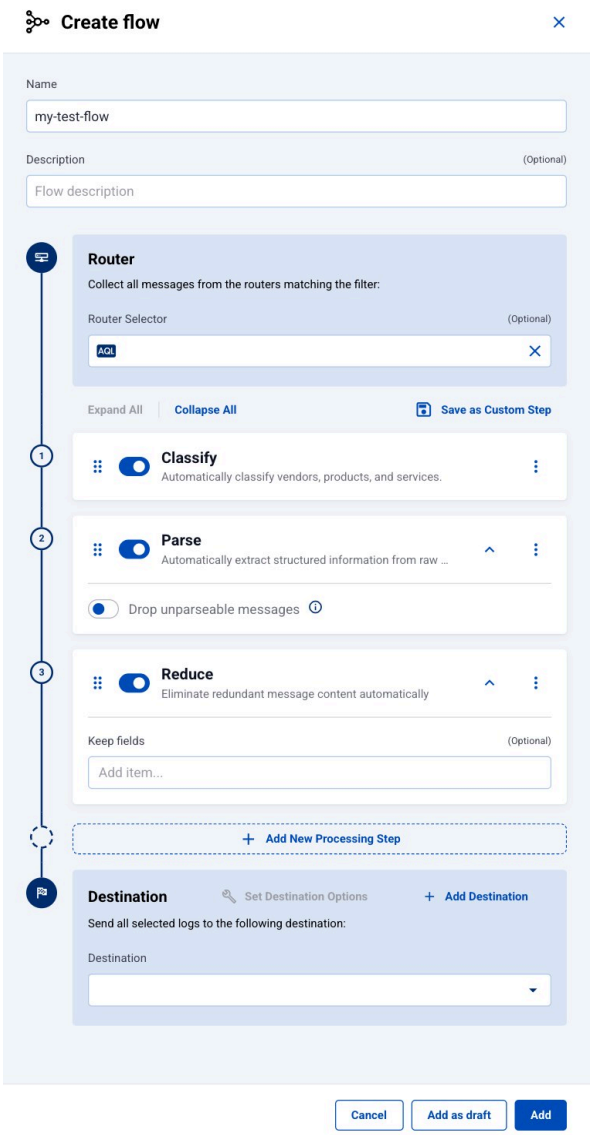
5. Select the **Destination** where you want to send your data. If you don't have any destination configured, you can select **Add Destination** in the destination section to create a new destination now. For details on the different destinations, see Destinations.

   - If you don't have any destination configured, see Destinations.
   - If you've already created a store, it's automatically available as a destination. Note that the **Router Selector** of the flow must match only AxoRouters that have the selected store available, otherwise you'll get an error message.
   - If you want to send data to another AxoRouter, enable the **Show all destinations** option, and select the connector of the AxoRouter where you want to send the data.



6. (Optional) To process the data transferred in the flow, select **Add New Processing Step**. For details, see Processing steps. For example:

   1. Add a **Classify**, a **Parse**, and a **Reduce** step, in that order, to automatically remove redundant and empty fields from your data.
   2. To select which messages are processed by the flow, add a **Select Messages** step, and enter a filter into the **AQL Expression** field. For example, to select only the messages received from Fortinet FortiGate firewalls, use the `meta.vendor = fortinet AND meta.product = fortigate` query.



7. Select **Add**, then confirm with **Add**. To save the flow without provisioning it to the routers yet, select **Add as draft** instead.

8. The new flow appears in the **Flows** list.



Protocol-specific destination options

If needed, select **More options** to set the following when using the TCP or TLS protocol. (**More options** also contains the **Disk buffer**, **Throttle**, and **Workers** options.)

- **TCP keepalive time interval**: The interval (number of seconds) between subsequential keepalive probes, regardless of the traffic exchanged in the connection.

- **TCP keepalive probes**: The number of unacknowledged probes to send before considering the connection dead.
- **TCP keepalive time**: The interval (in seconds) between the last data packet sent and the first keepalive probe.

Advanced options

- **Disk buffer**: Explicitly enable or disable disk buffer on AxoRouter. Default value: **Enabled**.

  AxoRouter can store messages on the local disk if the destination or the network connection to the destination becomes unavailable. AxoRouter automatically sends the stored messages (in the order they were received) to the destination when the connection is reestablished. Note that the disk buffer is separate from AxoStore.

  Disk buffer capacity (per worker): Size of the disk buffer files in bytes. This value applies for each worker. Default value is 100MiB (104857600 bytes), which is also the recommended minimum.

  The disk required for using disk buffering is **Disk buffer capacity (per worker)** * **Number of Workers** for each destination of AxoRouter.

Troubleshooting

If the data doesn't arrive at the destination, see the data doesn't reach the destination. To find the errors that the destination returns, see the destination rejects the data. For other problems, see Troubleshooting Axoflow.

8.9 Google

Send data from Axoflow to Google BigQuery, Google Cloud Pub/Sub, and Google Security Operations (SecOps).

8.9.1 BigQuery

Support for a Google BigQuery destination is coming soon. Until then, send data to BigQuery through Google Cloud Pub/Sub.

Axoflow will soon support sending data to Google BigQuery using a high-performance gRPC-based destination. Alternatively, you can set up AxoRouter to send data through a Google Pub/Sub destination.

If you'd like to send data from AxoRouter to this destination, contact our support team for details.

8.9.2 Google Cloud Pub/Sub

Add a Google Cloud Pub/Sub destination, and look up the Pub/Sub attributes and message fields it uses.

Google Cloud Pub/Sub: Asynchronous messaging service for ingesting and distributing event data between services and applications.

To add a Google Cloud Pub/Sub destination to Axoflow, complete the following steps. Axoflow can send data to Google Cloud Pub/Sub using its gRPC interface.

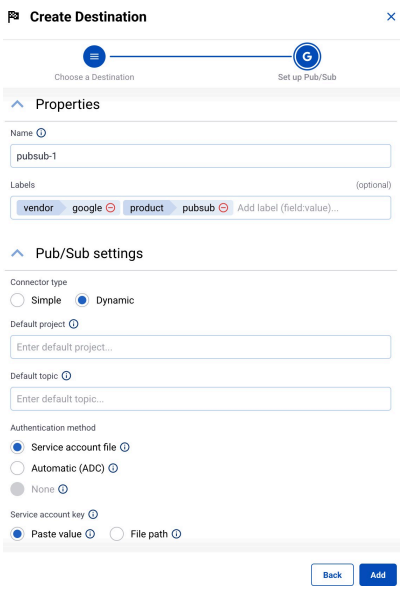
Prerequisites

- A Google Pub/Sub subscription.
- An IAM service account that Axoflow uses for authentication.
- A Google Cloud project that has the Pub/Sub API enabled.

For details, see the Google Pub/Sub tutorial.

Steps

1. Create a new destination.
    1. Open the AxoConsole.
    2. Select **Destinations > Add Destination > Destination**.
2. Configure the destination.
    1. Select **Pub/Sub**.
    2. Select which type of configuration you want to use:
        - **Simple**: Send all data into a project and topic.
        - **Dynamic**: Send data to a project and topic based on the content or metadata of the incoming messages (or to a default project/topic).
    3. Enter a name for the destination.



4. Specify the project and topic to send the data to.

   - **Simple**: Enter the ID of the GCP **Project** and the **Topic**. All data will be sent to this topic.
   - **Dynamic**: Enter the expression that specifies the default **Project** and **Topic**. The data will be sent here unless it is set during the processing of the message (for example, by the processing steps of the **Flow**).

   You can use AxoSyslog macros in this field.

5. Configure the authentication method to access the GCP project.

   - **Automatic (ADC)**: Use the service account attached to the cloud resource (VM) that hosts AxoRouter.
   - **Service Account File**: Specify the service account key.
     - To provide the key from the AxoConsole, select **Paste value** and paste the key. AxoConsole will automatically deploy the key to the related AxoRouters.
     - To provide the key from a file, select **File path** and enter the path where the service account key file is located (for example, `/etc/axorouter/user-config/serviceaccount.json`). The path must be mounted into the container, `/etc/axorouter` is mounted by default. You must manually copy that file to its place on every affected AxoRouter, or you can use the configuration management system of your choice.
   - **None**: Disable authentication completely. Only available when the **More options > Service Endpoint** option is set.

     CAUTION:

     Do not disable authentication in production.

6. (Optional) Set other options as needed for your environments.

   - **Service Endpoint**: Use a custom API endpoint. Leave it empty to use the default: `https://pubsub.googleapis.com`
   - **Disk buffer**: Explicitly enable or disable disk buffer on AxoRouter. Default value: **Enabled**.

     AxoRouter can store messages on the local disk if the destination or the network connection to the destination becomes unavailable. AxoRouter automatically sends the stored messages (in the order they were received) to the destination when the connection is reestablished. Note that the disk buffer is separate from AxoStore.

     Disk buffer capacity (per worker): Size of the disk buffer files in bytes. This value applies for each worker. Default value is 100MiB (104857600 bytes), which is also the recommended minimum.

     The disk required for using disk buffering is **Disk buffer capacity (per worker)** * **Number of Workers** for each destination of AxoRouter.

   - **Batch Size**: Sets the maximum size of payload in a batch. If the size of the messages reaches this value, AxoRouter sends the batch to the destination even if the number of messages is less than the value of the batch lines option.

     > **CAUTION:**
     >
     > When setting **Batch Size**:
     >
     > - Note that the actual HTTP request will be about 4KB larger because of the request headers.
     > - Check the configuration of your destination, as SaaS solutions and ingress controllers often limit the maximal size of HTTP requests. If the sum of **Batch Size** and the request headers exceeds this limit, the data will be rejected. (In case of NGINX, with a `client intended to send too large body` error message.)

   - **Batch Lines**: Number of lines sent to the destination in one batch. AxoRouter waits for this number of lines to accumulate and sends them off in a single batch. Increasing this number increases throughput as more messages are sent in a single batch, but also increases message latency.

     Note that **Initial window size** * **Maximum Connections** must be greater or equal than **Batch lines** * **Number of workers** for all the sources that feeds a given destination. Otherwise, the destination worker might be unable to send data until **Batch timeout** expires, even if there's data available.

For more details, see the AxoSyslog documentation.

- **Batch Timeout**: Maximal time in milliseconds to wait for a batch to be filled before sending it. The default value (`-1`) means that the batch should be full before sending, which may result in a long wait if the incoming traffic is low.

  For more details, see the AxoSyslog documentation.

- **Number of Workers**: Used for scaling the destination in case of high message load. Specifies the number of worker threads AxoRouter uses for sending messages to the destination. The default is 1. If high message throughput is not a concern, leave it on 1. For maximum performance, increase it up to the number of CPU cores available on AxoRouter.

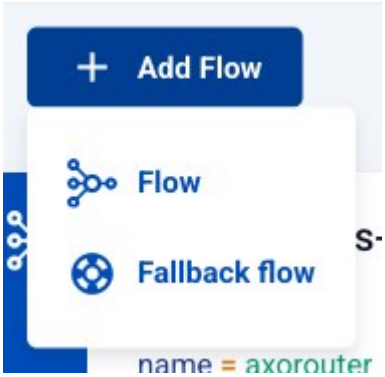
  For more details, see the AxoSyslog documentation.

7. Select **Add**.

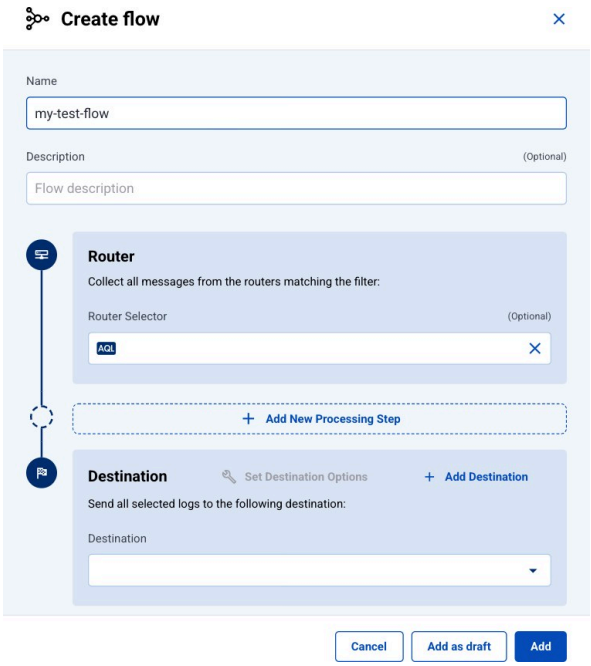
3. **Create a flow to connect the new destination to an AxoRouter instance.**

   1. Select **Flows**.
   2. Select **Add Flow > Flow**.

      To create a fallback flow, select **Add Flow > Fallback flow**.

   3. Enter a name for the flow, for example, `my-test-flow` .



4. In the **Router Selector** field, enter an expression that matches the routers you want to apply the flow to. To select a specific router, use a name selector, for example, `name = my-axorouter-hostname`.

   You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

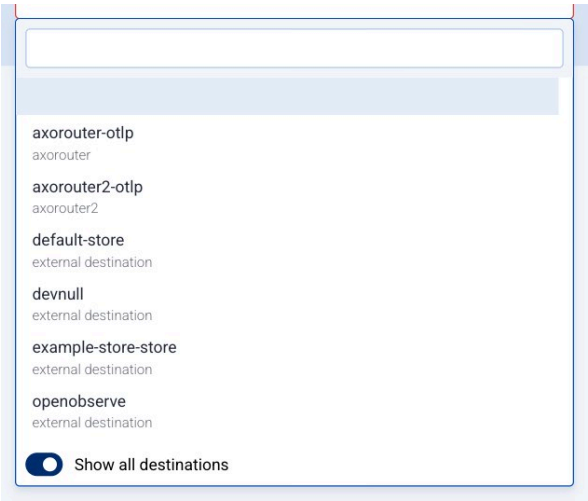
   - If you leave the router selector empty, it matches every AxoRouter instance.
   - To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.
   - If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

   > **Note**
   >
   > **You can configure multiple fallback flows, but only one fallback flow can apply to an AxoRouter (so the Router Selector of the fallback flows can't overlap).**

5. Select the **Destination** where you want to send your data. If you don't have any destination configured, you can select **Add Destination** in the destination section to create a new destination now. For details on the different destinations, see Destinations.

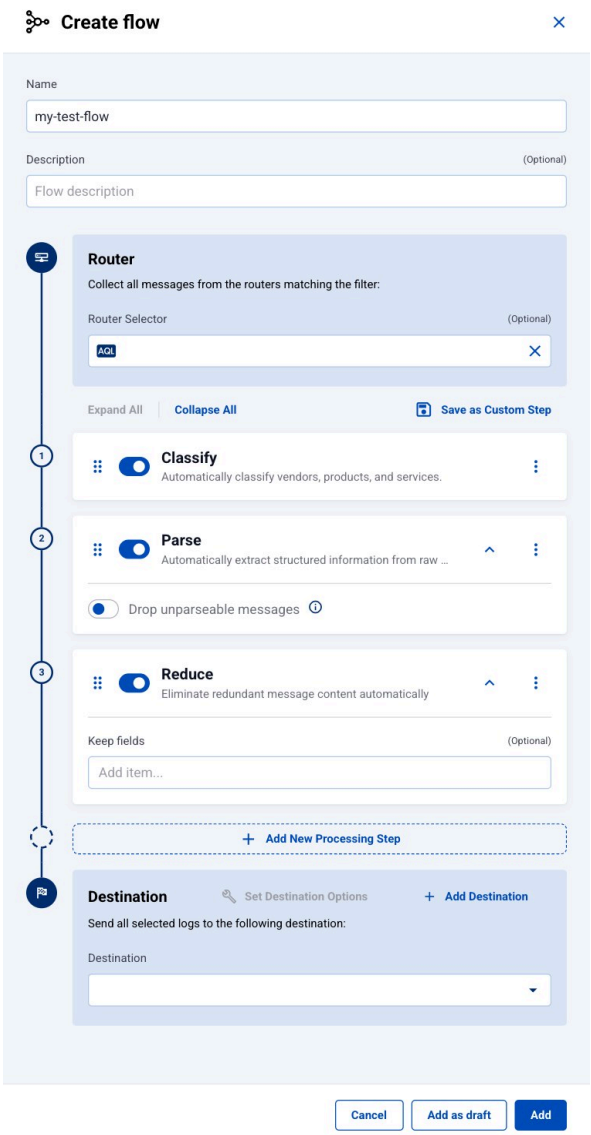
   - If you don't have any destination configured, see Destinations.

- If you've already created a store, it's automatically available as a destination. Note that the **Router Selector** of the flow must match only AxoRouters that have the selected store available, otherwise you'll get an error message.
- If you want to send data to another AxoRouter, enable the **Show all destinations** option, and select the connector of the AxoRouter where you want to send the data.



6. (Optional) To process the data transferred in the flow, select **Add New Processing Step**. For details, see Processing steps. For example:

   1. Add a **Classify**, a **Parse**, and a **Reduce** step, in that order, to automatically remove redundant and empty fields from your data.
   2. To select which messages are processed by the flow, add a **Select Messages** step, and enter a filter into the **AQL Expression** field. For example, to select only the messages received from Fortinet FortiGate firewalls, use the `meta.vendor = fortinet AND meta.product = fortigate` query.



7. Select **Add**, then confirm with **Add**. To save the flow without provisioning it to the routers yet, select **Add as draft** instead.

8. The new flow appears in the **Flows** list.



Pub/Sub attributes

You can embed custom attributes as metadata in Pub/Sub messages in the processing steps of the Flow that's sending data to the Pub/Sub destination.

To do that:

1. Add a FilterX processing step to the Flow.

2. Edit the **Statements** field of the processing step:

    1. Add the `meta.pubsub.attributes = json();` line to add an empty JSON object to the messages.

    2. Set your custom attributes under the `meta.pubsub.attributes` key. For example, if you want to include the timestamp as a custom attribute as well, you can use:

        ```
        meta.pubsub.attributes = {"timestamp": $S_ISODATE};
        ```

Related message fields

You can use the following message fields to modify messages sent to this destination using processing steps.

Troubleshooting

If the data doesn't arrive at the destination, see the data doesn't reach the destination. To find the errors that the destination returns, see the destination rejects the data. For other problems, see Troubleshooting Axoflow.

8.9.3 Google Security Operations (SecOps)

Add a Google Security Operations (SecOps) destination, look up the message fields it uses, and troubleshoot ingestion.

Google Security Operations (SecOps): A cloud service designed for enterprises to privately retain, analyze, and search the massive amounts of security and network telemetry they generate.

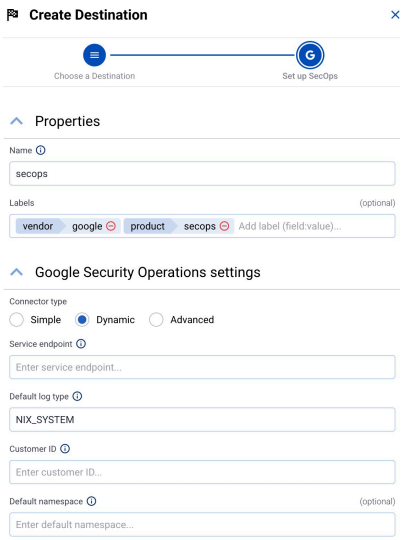
To add a Google Security Operations (SecOps) destination to Axoflow, complete the following steps.

Prerequisites

- A Google SecOps subscription.
- An IAM service account that Axoflow uses for authentication.

Steps

1. Create a new destination.
    1. Open the AxoConsole.
    2. Select **Destinations > Add Destination > Destination**.
2. Configure the destination.
    1. Select **SecOps**.
    2. Select which type of configuration you want to use:
        - **Simple**: Send all data into a namespace with a single log type.
        - **Dynamic**: Send data to a namespace with a log type based on the content or metadata of the incoming messages (or to a default namespace/log type).
        - **Advanced**: Specify the full endpoint URL.
    3. Enter a name for the destination.



4. Enter the URL endpoint for the destination.

   - **Simple** and **Dynamic**: Enter base URL for the regional service endpoint to use into the **Service Endpoint** field, for example, `https://malachiteingestion-pa.googleapis.com` for the US region.
   - **Advanced**: Enter the full URL to use for ingesting logs into the **Endpoint URL** field, for example, `https://malachiteingestion-pa.googleapis.com/v2/unstructuredlogentries:batchCreate` .

5. Specify the log type to send the data to.

   - **Simple** and **Advanced**: Enter the **Log type** to assign to the data. All data will be assigned the specified log type.
   - **Dynamic**: Enter the **Default log type**. The data will be sent with the specified log type unless it is set during the processing of the message (for example, by the automatic classification, or the processing steps of the **Flow**).

   You can use AxoSyslog macros in this field.

6. Enter the unique ID of your Google SecOps instance into the **Customer ID** field.

7. (Optional) Specify the namespace to send the data to. Usually it's not needed to use namespaces.

   - **Simple** and **Advanced**: Enter the **Namespace** where to send the data. All data will be sent to this namespace.
   - **Dynamic**: Enter the **Default namespace**. The data will be sent into this namespace unless it is set during the processing of the message (for example, by the processing steps of the **Flow**).

   You can use AxoSyslog macros in this field.

8. Configure the authentication method to access the GCP project.

- **Automatic (ADC)**: Use the service account attached to the cloud resource (VM) that hosts AxoRouter.
- **Service Account File**: Specify the service account key.
  - To provide the key from the AxoConsole, select **Paste value** and paste the key. AxoConsole will automatically deploy the key to the related AxoRouters.
  - To provide the key from a file, select **File path** and enter the path where the service account key file is located (for example, `/etc/axorouter/user-config/serviceaccount.json`). The path must be mounted into the container, `/etc/axorouter` is mounted by default. You must manually copy that file to its place on every affected AxoRouter, or you can use the configuration management system of your choice.

9. (Optional) Set other options as needed for your environments.

- **Timeout**: The number of seconds to wait for a log-forwarding request to complete, and attempt to reconnect the server if exceeded. If the timeout is exceeded, AxoRouter attempts to reconnect the destination. The default (`0`) is unlimited.

  For more details, see the AxoSyslog documentation.

- **Disk buffer**: Explicitly enable or disable disk buffer on AxoRouter. Default value: **Enabled**.

  AxoRouter can store messages on the local disk if the destination or the network connection to the destination becomes unavailable. AxoRouter automatically sends the stored messages (in the order they were received) to the destination when the connection is reestablished. Note that the disk buffer is separate from AxoStore.

  Disk buffer capacity (per worker): Size of the disk buffer files in bytes. This value applies for each worker. Default value is 100MiB (104857600 bytes), which is also the recommended minimum.

  The disk required for using disk buffering is **Disk buffer capacity (per worker)** * **Number of Workers** for each destination of AxoRouter.

- **Batch Size**: Sets the maximum size of payload in a batch. If the size of the messages reaches this value, AxoRouter sends the batch to the destination even if the number of messages is less than the value of the batch lines option.

  CAUTION:

When setting **Batch Size**:

- Note that the actual HTTP request will be about 4KB larger because of the request headers.
- Check the configuration of your destination, as SaaS solutions and ingress controllers often limit the maximal size of HTTP requests. If the sum of **Batch Size** and the request headers exceeds this limit, the data will be rejected. (In case of NGINX, with a `client intended to send too large body` error message.)

- **Batch Lines**: Number of lines sent to the destination in one batch. AxoRouter waits for this number of lines to accumulate and sends them off in a single batch. Increasing this number increases throughput as more messages are sent in a single batch, but also increases message latency.

  Note that **Initial window size** * **Maximum Connections** must be greater or equal than **Batch lines** * **Number of workers** for all the sources that feeds a given destination. Otherwise, the destination worker might be unable to send data until **Batch timeout** expires, even if there's data available.

  For more details, see the AxoSyslog documentation.

- **Batch Timeout**: Maximal time in milliseconds to wait for a batch to be filled before sending it. The default value (`-1`) means that the batch should be full before sending, which may result in a long wait if the incoming traffic is low.

  For more details, see the AxoSyslog documentation.

- **Content compression**: Specifies the compression to use when sending data to the destination. `No compression` explicitly disables compression. Other possible values: `Gzip` , `Deflate` . Select only compression that your destination supports and has enabled. Default value: `No compression`

- **Number of retries**: The number of retry attempts when the retry action is triggered. Default: `3` .

  For more details, see the AxoSyslog documentation.

- **Time to reopen**: The time to wait before reconnecting when a connection fails. Default: `1` second.

  For more details, see the AxoSyslog documentation.

- **Timeout**: The number of seconds to wait for a log-forwarding request to complete, and attempt to reconnect the server if exceeded. If the timeout is exceeded, AxoRouter attempts to reconnect the destination. The default (`0`) is unlimited.

For more details, see the AxoSyslog documentation.

- **Number of Workers**: Used for scaling the destination in case of high message load. Specifies the number of worker threads AxoRouter uses for sending messages to the destination. The default is 1. If high message throughput is not a concern, leave it on 1. For maximum performance, increase it up to the number of CPU cores available on AxoRouter.

  For more details, see the AxoSyslog documentation.

- **Response actions**: Specifies what AxoRouter does with the log message, based on the response code received from the destination.

  - If the status code begins with 2 (for example, `200`), AxoRouter assumes the message was successfully sent.
  - If you've configured a custom action in AxoConsole for the status code, AxoRouter performs the specified action.
  - If the status code appears in the default response actions table, AxoRouter performs the specified action.
  - For other status codes, AxoRouter disconnects.

  To add a custom response action, select **Add new action**, then enter the **HTTP status code** and the related **Action**. The following actions are available:

  - **Disconnect**: Keep trying to resend the message indefinitely.
  - **Drop**: Drop the message without trying to resend it.
  - **Retry**: Retry sending the message for a maximum of 3 times.
  - **Success**: Assume the message was successfully sent.

Show default actions

code	explanation	action
100	"Continue"	disconnect
101	"Switching Protocols"	disconnect
102	"Processing"	retry
103	"Early Hints"	retry
200	"OK"	success
201	"Created"	success
202	"Accepted"	success

203	“Non-Authoritative Information”	success
204	“No Content”	success
205	“Reset Content”	success
206	“Partial Content”	success
300	“Multiple Choices”	disconnect
301	“Moved Permanently”	disconnect
302	“Found”	disconnect
303	“See Other”	disconnect
304	“Not Modified”	retry
307	“Temporary Redirect”	disconnect
308	“Permanent Redirect”	disconnect
400	“Bad Request”	disconnect
401	“Unauthorized”	disconnect
402	“Payment Required”	disconnect
403	“Forbidden”	disconnect
404	“Not Found”	disconnect
405	“Method Not Allowed”	disconnect
406	“Not Acceptable”	disconnect
407	“Proxy Authentication Required”	disconnect
408	“Request Timeout”	disconnect
409	“Conflict”	disconnect
410	“Gone”	drop
411	“Length Required”	disconnect
412	“Precondition Failed”	disconnect
413	“Payload Too Large”	disconnect

414	“URI Too Long”	disconnect
415	“Unsupported Media Type”	disconnect
416	“Range Not Satisfiable”	drop
417	“Expectation Failed”	disconnect
418	“I’m a teapot”	disconnect
421	“Misdirected Request”	disconnect
422	“Unprocessable Entity”	drop
423	“Locked”	disconnect
424	“Failed Dependency”	drop
425	“Too Early”	drop
426	“Upgrade Required”	disconnect
428	“Precondition Required”	retry
429	“Too Many Requests”	disconnect
431	“Request Header Fields Too Large”	disconnect
451	“Unavailable For Legal Reasons”	drop
500	“Internal Server Error”	disconnect
501	“Not Implemented”	disconnect
502	“Bad Gateway”	disconnect
503	“Service Unavailable”	disconnect
504	“Gateway Timeout”	retry
505	“HTTP Version Not Supported”	disconnect
506	“Variant Also Negotiates”	disconnect
507	“Insufficient Storage”	disconnect
508	“Loop Detected”	drop
510	“Not Extended”	disconnect

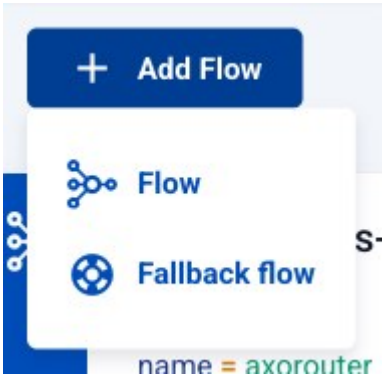
511	“Network Authentication Required”	disconnect

10. Select **Add**.

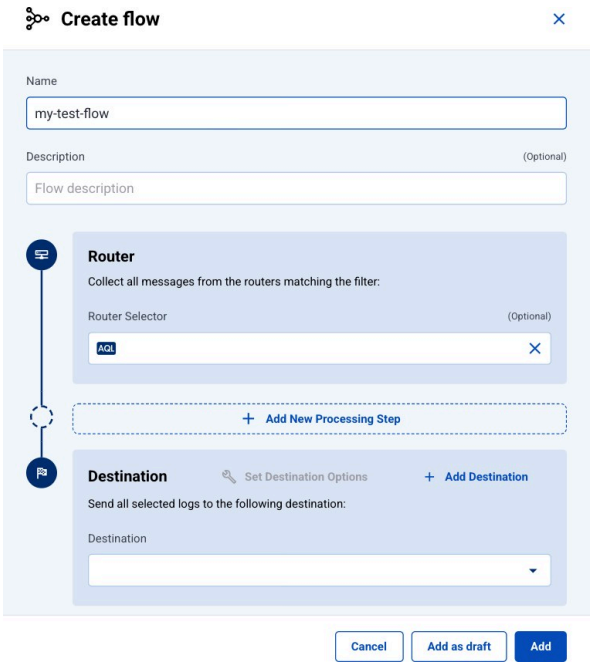
3. **Create a flow to connect the new destination to an AxoRouter instance.**

    1. Select **Flows**.
    2. Select **Add Flow > Flow**.

       To create a fallback flow, select **Add Flow > Fallback flow**.

    3. Enter a name for the flow, for example, `my-test-flow`.

    4. In the **Router Selector** field, enter an expression that matches the routers you want to apply the flow to. To select a specific router, use a name selector, for example, `name = my-axorouter-hostname`.

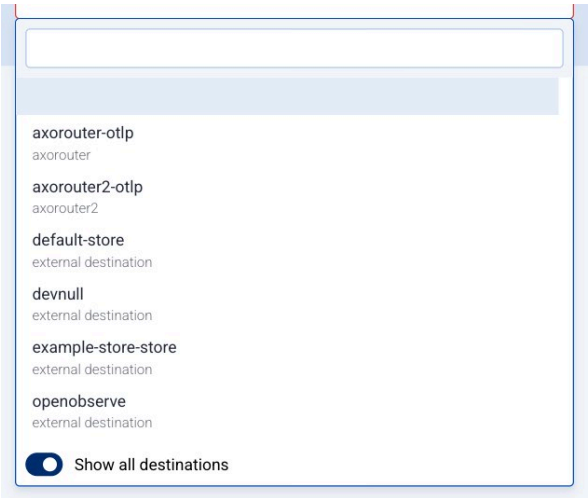
       You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

- If you leave the router selector empty, it matches every AxoRouter instance.
- To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.
- If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

> **Note**
>
> **You can configure multiple fallback flows, but only one fallback flow can apply to an AxoRouter (so the Router Selector of the fallback flows can't overlap).**

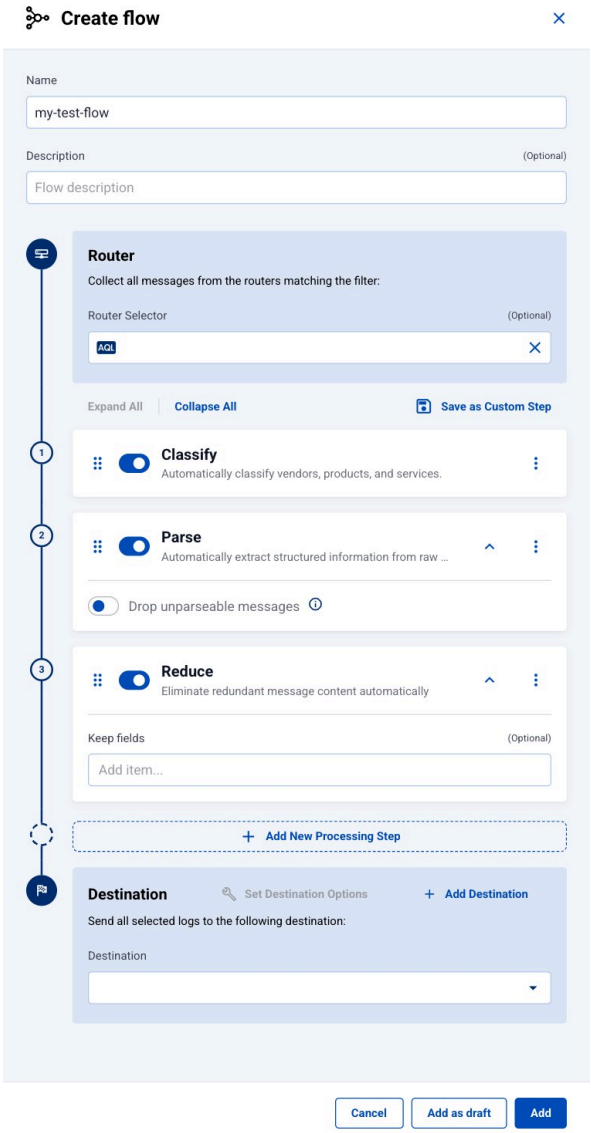
5. Select the **Destination** where you want to send your data. If you don't have any destination configured, you can select **Add Destination** in the destination section to create a new destination now. For details on the different destinations, see Destinations.

   - If you don't have any destination configured, see Destinations.
   - If you've already created a store, it's automatically available as a destination. Note that the **Router Selector** of the flow must match only AxoRouters that have the selected store available, otherwise you'll get an error message.
   - If you want to send data to another AxoRouter, enable the **Show all destinations** option, and select the connector of the AxoRouter where you want to send the data.



6. (Optional) To process the data transferred in the flow, select **Add New Processing Step**. For details, see Processing steps. For example:

1. Add a **Classify**, a **Parse**, and a **Reduce** step, in that order, to automatically remove redundant and empty fields from your data.
2. To select which messages are processed by the flow, add a **Select Messages** step, and enter a filter into the **AQL Expression** field. For example, to select only the messages received from Fortinet FortiGate firewalls, use the `meta.vendor = fortinet AND meta.product = fortigate` query.



7. Select **Add**, then confirm with **Add**. To save the flow without provisioning it to the routers yet, select **Add as draft** instead.
8. The new flow appears in the **Flows** list.

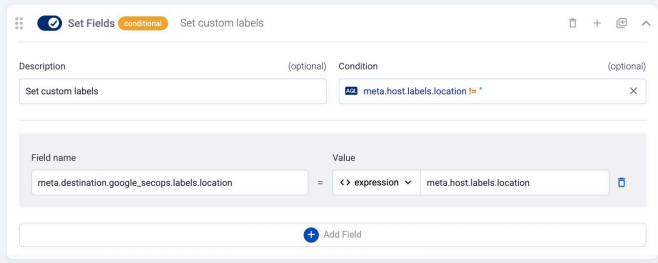


4. If your logs don't start to show up in Google SecOps, check the Troubleshooting section.

Related message fields

You can use the following message fields to modify messages sent to this destination using processing steps.

For example, you can add your custom labels using a **Set Fields** processing step to `meta.destination.google_secops.labels`:



Note that sending custom labels to SecOps can have significant performance impact. Make sure to increase the number of workers (**More options > Number of Workers**), ideally to have at least one worker for each custom label.

Troubleshooting

If your logs aren't showing up in Google SecOps, tap the service logs of the axorouter-syslog service on an AxoRouter that's sending data to Google SecOps.

- If you see errors like `http: Server returned with a 4XX (client errors) status code, which means we are not authorized or the URL is not found;`, double-check the **Service endpoint** (make sure that it points to the right regional service endpoint) and **Customer ID** fields.

If the data doesn't arrive at the destination, see the data doesn't reach the destination. To find the errors that the destination returns, see the destination rejects the data. For other problems, see Troubleshooting Axoflow.

8.10 Grafana

Send log data from Axoflow to Grafana Loki for storage and querying.

8.10.1 Grafana Loki

Support for a Grafana Loki destination is coming soon. Request it, or ask about the status of the integration.

Grafana Loki: Scalable log aggregation system optimized for storing and querying logs by labels using Grafana.

If you'd like to send data from AxoRouter to this destination, contact our support team for details.

8.11 Microsoft

Send data from Axoflow to Microsoft Sentinel and Azure Monitor.

8.11.1 Azure Monitor

Send data to Azure Monitor with the same procedure as the Microsoft Sentinel destination.

Sending data to Azure Monitor is practically identical to using the Sentinel destination. Follow the procedure described in Microsoft Sentinel.

8.11.2 Microsoft Sentinel

Add a Microsoft Sentinel or Azure Monitor destination that sends data to built-in or custom tables, and look up its message fields.

Microsoft Sentinel: Cloud-native SIEM and SOAR platform for collecting, analyzing, and responding to security threats.

To add a Microsoft Sentinel or Azure Monitor destination to Axoflow, complete the following steps. AxoConsole can configure your AxoRouters to send data to the built-in **CommonSecurityLog**, **SecurityEvents**, **Syslog**, **WindowsEvents** tables of Azure Monitor. To send data to custom tables, contact our support team.

Prerequisites

The following list is a quick overview of the things you need to configure in Sentinel. For detailed instructions, see Configure Sentinel.

1. An Azure subscription.
2. An Enterprise application registration and client secret. You'll need the Tenant ID, Application ID, and Application Secret of the application to configure the Axoflow destination.
   - Entra ID should be configured to allow users to create new application registrations, otherwise Global or Application Administrator role is needed.
3. A Log Analytics Workspace in Azure.
   - Permissions required to create a workspace.
4. A Data Collection Endpoint (DCE) and a Data Collection Rule (DCR). We provide you with ARM templates for both the DCE and the DCR.
   - Permissions required to create and manage DCRs.
5. Permissions assigned to the DCR.
6. A Microsoft Sentinel workspace (optional).

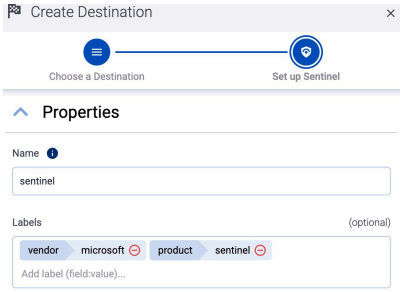
Steps

1. Create a new destination.

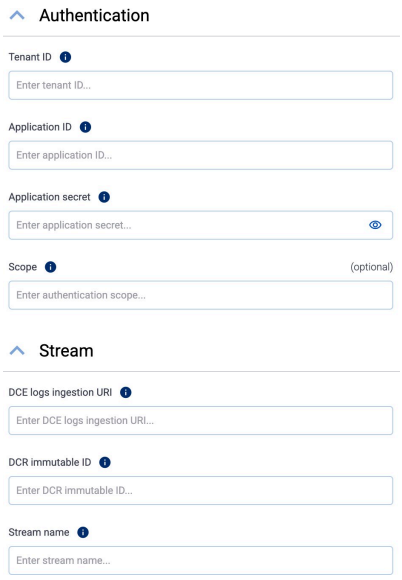
   1. Open the AxoConsole.
   2. Select **Destinations > Add Destination > Destination**.
2. Configure the destination.

   1. Select **Sentinel** or **Azure Monitor**.

2. Enter a name for the destination.



3. Configure the credentials needed for authentication.



   - **Tenant ID**: Directory (tenant) ID of the environment where you're sending the data. (Practically everything belongs to the tenant ID: the Entra ID application, the Log analytics workspace, Sentinel, the DCE and the DCR, and so on.)
   - **Application ID**: Application (client) ID of the Microsoft Entra ID application.
   - **Application secret**: The Client secret of the Microsoft Entra ID application.
   - **Scope**: The scope for the authentication token. Usually you can leave empty to use the default value (`https://monitor.azure.com//.default`).
4. Specify the details of the table to send the data to.
   - **DCE Logs Ingestion URI**: The URI of your Data Collection Endpoint (DCE).
   - **DCR Immutable ID**: The immutable ID property of the Azure Monitor Data Collection Rule (DCR) where AxoRouter sends the data.
   - **Stream name**: The name of the stream where AxoRouter sends the data. Use `Custom-Syslog-Stream` if you are in doubt. It is available in the provided templates and used as the default normalization scheme.
5. Select **More options**, and set the batch size for sending data to Sentinel. Recommended batch configuration:

- **Batch size**: By default, this is set to `960000` (960kB). Don't increase it over 1000000 to make sure the request body is smaller than the 1MB limit in Sentinel.
- **Batch timeout**: Set 10000 (10s) to control the delay of log ingestion.

6. (Optional) Set other options as needed for your environments.

   - **Timeout**: The number of seconds to wait for a log-forwarding request to complete, and attempt to reconnect the server if exceeded. If the timeout is exceeded, AxoRouter attempts to reconnect the destination. The default (`0`) is unlimited.

     For more details, see the AxoSyslog documentation.

   - **Disk buffer**: Explicitly enable or disable disk buffer on AxoRouter. Default value: **Enabled**.

     AxoRouter can store messages on the local disk if the destination or the network connection to the destination becomes unavailable. AxoRouter automatically sends the stored messages (in the order they were received) to the destination when the connection is reestablished. Note that the disk buffer is separate from AxoStore.

     Disk buffer capacity (per worker): Size of the disk buffer files in bytes. This value applies for each worker. Default value is 100MiB (104857600 bytes), which is also the recommended minimum.

     The disk required for using disk buffering is **Disk buffer capacity (per worker)** * **Number of Workers** for each destination of AxoRouter.

   - **Batch Size**: Sets the maximum size of payload in a batch. If the size of the messages reaches this value, AxoRouter sends the batch to the destination even if the number of messages is less than the value of the batch lines option.

     > **CAUTION:**
     >
     > When setting **Batch Size**:
     >
     > - Note that the actual HTTP request will be about 4KB larger because of the request headers.
     > - Check the configuration of your destination, as SaaS solutions and ingress controllers often limit the maximal size of HTTP requests. If the sum of **Batch Size** and the request headers exceeds this limit, the data will be rejected. (In case of NGINX, with a `client intended to send too large body` error message.)

- **Batch Lines**: Number of lines sent to the destination in one batch. AxoRouter waits for this number of lines to accumulate and sends them off in a single batch. Increasing this number increases throughput as more messages are sent in a single batch, but also increases message latency.

  Note that **Initial window size** * **Maximum Connections** must be greater or equal than **Batch lines** * **Number of workers** for all the sources that feeds a given destination. Otherwise, the destination worker might be unable to send data until **Batch timeout** expires, even if there's data available.

  For more details, see the AxoSyslog documentation.

- **Batch Timeout**: Maximal time in milliseconds to wait for a batch to be filled before sending it. The default value (`-1`) means that the batch should be full before sending, which may result in a long wait if the incoming traffic is low.

  For more details, see the AxoSyslog documentation.

- **Content compression**: Specifies the compression to use when sending data to the destination. `No compression` explicitly disables compression. Other possible values: `Gzip` , `Deflate` . Select only compression that your destination supports and has enabled. Default value: `No compression`

- **Number of retries**: The number of retry attempts when the retry action is triggered. Default: `3` .

  For more details, see the AxoSyslog documentation.

- **Time to reopen**: The time to wait before reconnecting when a connection fails. Default: `1` second.

  For more details, see the AxoSyslog documentation.

- **Timeout**: The number of seconds to wait for a log-forwarding request to complete, and attempt to reconnect the server if exceeded. If the timeout is exceeded, AxoRouter attempts to reconnect the destination. The default (`0`) is unlimited.

  For more details, see the AxoSyslog documentation.

- **Number of Workers**: Used for scaling the destination in case of high message load. Specifies the number of worker threads AxoRouter uses for sending messages to the destination. The default is 1. If high message throughput is not a concern, leave it on 1. For maximum performance, increase it up to the number of CPU cores available on AxoRouter.

  For more details, see the AxoSyslog documentation.

- **Response actions**: Specifies what AxoRouter does with the log message, based on the response code received from the destination.
  - If the status code begins with 2 (for example, `200`), AxoRouter assumes the message was successfully sent.
  - If you've configured a custom action in AxoConsole for the status code, AxoRouter performs the specified action.
  - If the status code appears in the default response actions table, AxoRouter performs the specified action.
  - For other status codes, AxoRouter disconnects.

  To add a custom response action, select **Add new action**, then enter the **HTTP status code** and the related **Action**. The following actions are available:

  - **Disconnect**: Keep trying to resend the message indefinitely.
  - **Drop**: Drop the message without trying to resend it.
  - **Retry**: Retry sending the message for a maximum of 3 times.
  - **Success**: Assume the message was successfully sent.

Show default actions

code	explanation	action
100	"Continue"	disconnect
101	"Switching Protocols"	disconnect
102	"Processing"	retry
103	"Early Hints"	retry
200	"OK"	success
201	"Created"	success
202	"Accepted"	success
203	"Non-Authoritative Information"	success
204	"No Content"	success
205	"Reset Content"	success
206	"Partial Content"	success
300	"Multiple Choices"	disconnect
301	"Moved Permanently"	disconnect

302	“Found”	disconnect
303	“See Other”	disconnect
304	“Not Modified”	retry
307	“Temporary Redirect”	disconnect
308	“Permanent Redirect”	disconnect
400	“Bad Request”	disconnect
401	“Unauthorized”	disconnect
402	“Payment Required”	disconnect
403	“Forbidden”	disconnect
404	“Not Found”	disconnect
405	“Method Not Allowed”	disconnect
406	“Not Acceptable”	disconnect
407	“Proxy Authentication Required”	disconnect
408	“Request Timeout”	disconnect
409	“Conflict”	disconnect
410	“Gone”	drop
411	“Length Required”	disconnect
412	“Precondition Failed”	disconnect
413	“Payload Too Large”	disconnect
414	“URI Too Long”	disconnect
415	“Unsupported Media Type”	disconnect
416	“Range Not Satisfiable”	drop
417	“Expectation Failed”	disconnect
418	“I’m a teapot”	disconnect
421	“Misdirected Request”	disconnect

422	“Unprocessable Entity”	drop
423	“Locked”	disconnect
424	“Failed Dependency”	drop
425	“Too Early”	drop
426	“Upgrade Required”	disconnect
428	“Precondition Required”	retry
429	“Too Many Requests”	disconnect
431	“Request Header Fields Too Large”	disconnect
451	“Unavailable For Legal Reasons”	drop
500	“Internal Server Error”	disconnect
501	“Not Implemented”	disconnect
502	“Bad Gateway”	disconnect
503	“Service Unavailable”	disconnect
504	“Gateway Timeout”	retry
505	“HTTP Version Not Supported”	disconnect
506	“Variant Also Negotiates”	disconnect
507	“Insufficient Storage”	disconnect
508	“Loop Detected”	drop
510	“Not Extended”	disconnect
511	“Network Authentication Required”	disconnect

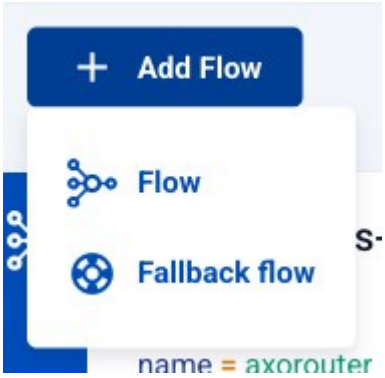
7. Select **Add**.

3. **Create a flow to connect the new destination to an AxoRouter instance.**

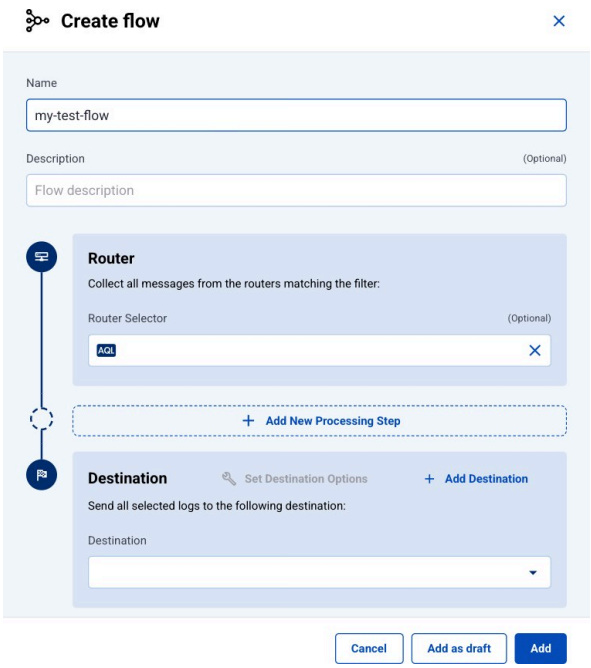
   1. Select **Flows**.

   2. Select **Add Flow > Flow**.

      To create a fallback flow, select **Add Flow > Fallback flow**.



3. Enter a name for the flow, for example, `my-test-flow`.



4. In the **Router Selector** field, enter an expression that matches the routers you want to apply the flow to. To select a specific router, use a name selector, for example, `name = my-axorouter-hostname`.

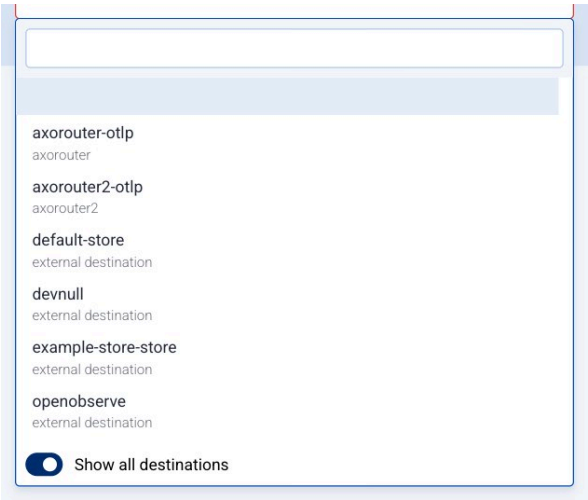
   You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

   - If you leave the router selector empty, it matches every AxoRouter instance.
   - To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.
   - If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

> **Note**
>
> **You can configure multiple fallback flows, but only one fallback flow can apply to an AxoRouter (so the Router Selector of the fallback flows can't overlap).**

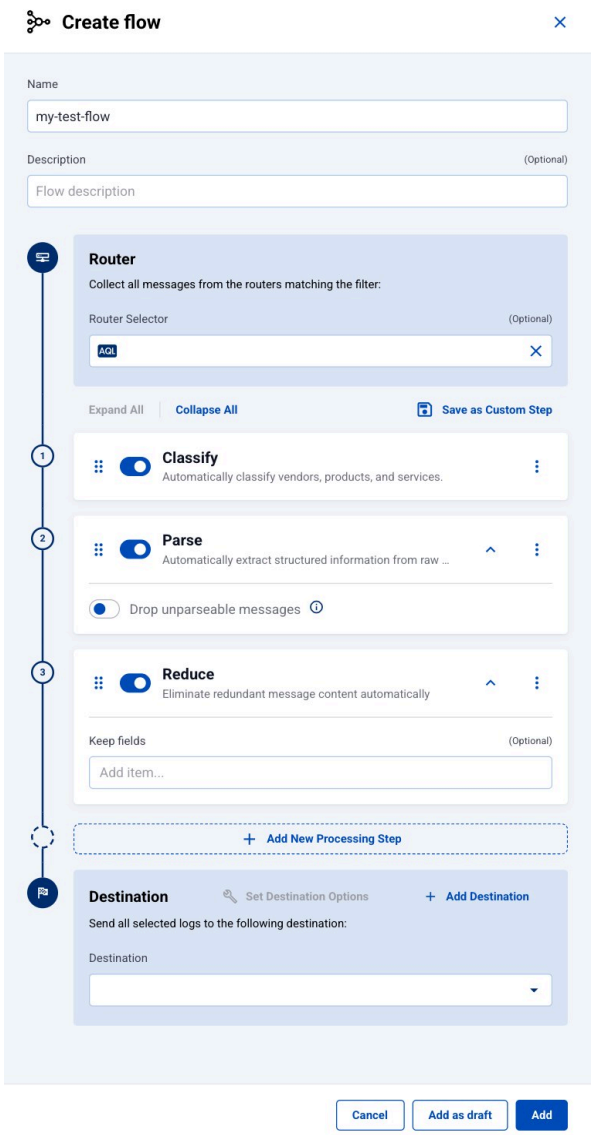
5. Select the **Destination** where you want to send your data. If you don't have any destination configured, you can select **Add Destination** in the destination section to create a new destination now. For details on the different destinations, see Destinations.

   - If you don't have any destination configured, see Destinations.
   - If you've already created a store, it's automatically available as a destination. Note that the **Router Selector** of the flow must match only AxoRouters that have the selected store available, otherwise you'll get an error message.
   - If you want to send data to another AxoRouter, enable the **Show all destinations** option, and select the connector of the AxoRouter where you want to send the data.



6. (Optional) To process the data transferred in the flow, select **Add New Processing Step**. For details, see Processing steps. For example:

   1. Add a **Classify**, a **Parse**, and a **Reduce** step, in that order, to automatically remove redundant and empty fields from your data.
   2. To select which messages are processed by the flow, add a **Select Messages** step, and enter a filter into the **AQL Expression** field. For example, to select only the messages received from Fortinet FortiGate firewalls, use the `meta.vendor = fortinet AND meta.product = fortigate` query.



7. Select **Add**, then confirm with **Add**. To save the flow without provisioning it to the routers yet, select **Add as draft** instead.

8. The new flow appears in the **Flows** list.



Related message fields

You can use the following message fields to modify messages sent to this destination using processing steps.

Troubleshooting

If the data doesn't arrive at the destination, see the data doesn't reach the destination. To find the errors that the destination returns, see the destination rejects the data. For other problems, see Troubleshooting Axoflow.

8.11.2.1 Configure Sentinel

Create the Azure application, workspace, data collection endpoint and rule, and permissions that AxoRouter needs to send data to Sentinel.

To send data from AxoRouter to Microsoft Sentinel, you have to configure a number of things in Sentinel before configuring a destination in AxoConsole. Complete the following steps.

Create an Azure App

Create an Azure (Microsoft Entra) application and credentials for it.

1. Navigate to **App registrations** on the Azure Portal.
2. Click **New registration** and register an application (for example `AxoflowIngestion`).
3. Save the **Application (Client) ID** and the **Directory (Tenant) ID** (you'll need these later to configure Axoflow).
4. Go to **Certificates & secrets > Client secrets > + New client secret**.
5. Add a description and expiration, click **Add**, and record the **secret value** (OAuth secret). You'll need this later to configure Axoflow.

   NOTE: Set a reminder to renew the secret and update your Axoflow configuration before it expires, otherwise your AxoRouters won't be able to send data to Sentinel, possibly causing data loss.

Enable Microsoft Sentinel on a Log Analytics Workspace

1. In the Azure Portal search for **Microsoft Sentinel**.
2. Select an existing workspace or create a new one (choose **Resource Group** and **Region**).
3. Once added, select the workspace and open **Tables**.
4. You should see the **CommonSecurityLog**, **SecurityEvents**, **Syslog**, **WindowsEvents** built-in tables. Sometimes the **Syslog** table appears only when it has data.
5. From the workspace **Overview**, open **JSON View** and record the **Workspace Resource ID** (needed in templates).

Create a Data Collection Endpoint (DCE)

To create a Data Collection Endpoint, complete the following steps. For more details, see the Microsoft Sentinel documentation.

1. Search for **Deploy a custom template** in Azure services.
2. Choose **Build your own template** in the editor.
3. Download the Axoflow DCE template, and upload or paste it into the template editor. Click **Save**.
4. Set the parameters of the DCE. Make sure **Region** matches your **Sentinel workspace**.
5. After creation, open it and copy its endpoint URL (that's `logsIngestion.endpoint` in the JSON view) and its **Resource ID** (`id`). You'll need the endpoint later to configure Axoflow, and the resource ID to configure the data collection rule.

Create a Data Collection Rule (DCR)

To create a Data Collection Rule, complete the following steps. For more details, see the Microsoft Sentinel documentation.

1. Search for **Deploy a custom template** in Azure services.
2. Choose **Build your own template** in the editor.
3. Download the Axoflow DCR template, and upload or paste it into the template editor. Click **Save**.
4. Set the parameters of the DCR. Enter the **Workspace Resource ID** and **Endpoint Resource ID** from the previous steps.
5. After creation, open it and copy its `immutableID` from the JSON view. You'll need it later to configure Axoflow.

Assign Permissions on the DCR

1. Open **Access control (IAM)** on the DCR.
2. Add a role assignment:
   - **Role**: Monitoring Metrics Publisher
   - **Member**: the Azure App you created in step 1 (Create an Azure App)
3. Review and assign. This gives the app permissions to push data into Sentinel via the DCR.

After configuring everything in Sentinel, configure a Sentinel destination in AxoConsole.

8.12 OpenObserve

Add an OpenObserve destination to send your security data to OpenObserve, and look up the message fields it uses.

OpenObserve: High-performance log analytics platform supporting structured log ingestion, search, and visualization.

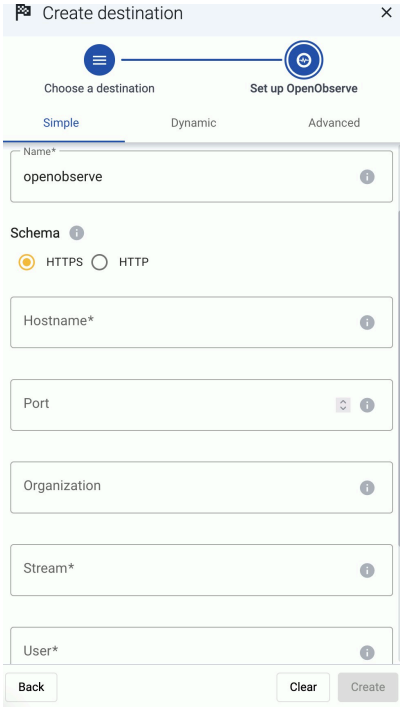
To add an OpenObserve destination to Axoflow, complete the following steps.

Prerequisites

- An OpenObserve account for Axoflow, or
- a self-hosted OpenObserve deployment.
- To configure Axoflow, you'll need the username, password, the name of your organization, and the name of the OpenObserve stream where you want to send your data.

Steps

1. Create a new destination.
    1. Open the AxoConsole.
    2. Select **Destinations > Add Destination > Destination**.
2. Configure the destination.
    1. Select **OpenObserve**.
    2. Select which type of configuration you want to use:
        - **Simple**: Send all data into a single stream of an organization.
        - **Dynamic**: Send data to an organization and stream based on the content or metadata of the incoming messages (or to the default stream of the default organization).
        - **Advanced**: Allows you to specify a custom URL endpoint.
    3. Enter a name for the destination.



4. Configure the endpoint of the destination.

   - **Advanced**: Enter your OpenObserve URL into the **URL** field, for example, `https://example.com/api/my-org/my-stream/_json`
   - **Simple** and **Dynamic**:
     1. Select the **HTTPS** or **HTTP** protocol to use to access your destination.
     2. Enter the **Hostname** and **Port** of the destination.

5. Specify the OpenObserve stream to send the data to.

   - **Simple**: Enter name of the **Organization** and the **Stream** where you want to send the data. All data will be sent into this stream.
   - **Dynamic**: Enter the expression that specifies the **Default organization** and the **Default stream**. The data will be sent into this stream if no other organization and stream is set during the processing of the message (for example, by the processing steps of the **Flow**).

   You can use AxoSyslog macros in this field.

6. Enter the username and password for the account you want to use.

7. (Optional)

   By default, Axoflow rejects connections to the destination server if the certificate of the server is invalid (for example, it's expired, signed by an unknown CA, or its CN and the name of the server is mismatched: the `Common Name` or the `subject_alt_name` parameter of the peer's certificate must contain the hostname or the IP address (as resolved from AxoRouter) of the peer).

> **Warning**
>
> **When validating a certificate, the entire certificate chain must be valid, including the CA certificate. If any certificate of the chain is invalid, Axoflow will reject the connection.**

If you want to accept invalid certificates (or no certificate) from the destination servers, disable the **Verify server certificate** option.

8. (Optional) Set other options as needed for your environments.

   - **Timeout**: The number of seconds to wait for a log-forwarding request to complete, and attempt to reconnect the server if exceeded. If the timeout is exceeded, AxoRouter attempts to reconnect the destination. The default (`0`) is unlimited.

     For more details, see the AxoSyslog documentation.

   - **Disk buffer**: Explicitly enable or disable disk buffer on AxoRouter. Default value: **Enabled**.

     AxoRouter can store messages on the local disk if the destination or the network connection to the destination becomes unavailable. AxoRouter automatically sends the stored messages (in the order they were received) to the destination when the connection is reestablished. Note that the disk buffer is separate from AxoStore.

     Disk buffer capacity (per worker): Size of the disk buffer files in bytes. This value applies for each worker. Default value is 100MiB (104857600 bytes), which is also the recommended minimum.

     The disk required for using disk buffering is **Disk buffer capacity (per worker)** * **Number of Workers** for each destination of AxoRouter.

   - **Batch Size**: Sets the maximum size of payload in a batch. If the size of the messages reaches this value, AxoRouter sends the batch to the destination even if the number of messages is less than the value of the batch lines option.

     > **CAUTION:**
     >
     > When setting **Batch Size**:
     >
     > - Note that the actual HTTP request will be about 4KB larger because of the request headers.

- Check the configuration of your destination, as SaaS solutions and ingress controllers often limit the maximal size of HTTP requests. If the sum of **Batch Size** and the request headers exceeds this limit, the data will be rejected. (In case of NGINX, with a `client intended to send too large body` error message.)

- **Batch Lines**: Number of lines sent to the destination in one batch. AxoRouter waits for this number of lines to accumulate and sends them off in a single batch. Increasing this number increases throughput as more messages are sent in a single batch, but also increases message latency.

  Note that **Initial window size** * **Maximum Connections** must be greater or equal than **Batch lines** * **Number of workers** for all the sources that feeds a given destination. Otherwise, the destination worker might be unable to send data until **Batch timeout** expires, even if there's data available.

  For more details, see the AxoSyslog documentation.

- **Batch Timeout**: Maximal time in milliseconds to wait for a batch to be filled before sending it. The default value (`-1`) means that the batch should be full before sending, which may result in a long wait if the incoming traffic is low.

  For more details, see the AxoSyslog documentation.

- **Content compression**: Specifies the compression to use when sending data to the destination. `No compression` explicitly disables compression. Other possible values: `Gzip` , `Deflate` . Select only compression that your destination supports and has enabled. Default value: `No compression`

- **Number of retries**: The number of retry attempts when the retry action is triggered. Default: `3` .

  For more details, see the AxoSyslog documentation.

- **Time to reopen**: The time to wait before reconnecting when a connection fails. Default: `1` second.

  For more details, see the AxoSyslog documentation.

- **Timeout**: The number of seconds to wait for a log-forwarding request to complete, and attempt to reconnect the server if exceeded. If the timeout is exceeded, AxoRouter attempts to reconnect the destination. The default (`0`) is unlimited.

  For more details, see the AxoSyslog documentation.

- **Number of Workers**: Used for scaling the destination in case of high message load. Specifies the number of worker threads AxoRouter uses for sending messages to the destination. The default is 1. If high message throughput is not a concern, leave it on 1. For maximum performance, increase it up to the number of CPU cores available on AxoRouter.

  For more details, see the AxoSyslog documentation.

- **Response actions**: Specifies what AxoRouter does with the log message, based on the response code received from the destination.

  - If the status code begins with 2 (for example, `200`), AxoRouter assumes the message was successfully sent.
  - If you've configured a custom action in AxoConsole for the status code, AxoRouter performs the specified action.
  - If the status code appears in the default response actions table, AxoRouter performs the specified action.
  - For other status codes, AxoRouter disconnects.

  To add a custom response action, select **Add new action**, then enter the **HTTP status code** and the related **Action**. The following actions are available:

  - **Disconnect**: Keep trying to resend the message indefinitely.
  - **Drop**: Drop the message without trying to resend it.
  - **Retry**: Retry sending the message for a maximum of 3 times.
  - **Success**: Assume the message was successfully sent.

> **Show default actions**
>
> | code | explanation | action |
> |---|---|---|
> | 100 | "Continue" | disconnect |
> | 101 | "Switching Protocols" | disconnect |
> | 102 | "Processing" | retry |
> | 103 | "Early Hints" | retry |
> | 200 | "OK" | success |
> | 201 | "Created" | success |
> | 202 | "Accepted" | success |
> | 203 | "Non-Authoritative Information" | success |

204	“No Content”	success
205	“Reset Content”	success
206	“Partial Content”	success
300	“Multiple Choices”	disconnect
301	“Moved Permanently”	disconnect
302	“Found”	disconnect
303	“See Other”	disconnect
304	“Not Modified”	retry
307	“Temporary Redirect”	disconnect
308	“Permanent Redirect”	disconnect
400	“Bad Request”	disconnect
401	“Unauthorized”	disconnect
402	“Payment Required”	disconnect
403	“Forbidden”	disconnect
404	“Not Found”	disconnect
405	“Method Not Allowed”	disconnect
406	“Not Acceptable”	disconnect
407	“Proxy Authentication Required”	disconnect
408	“Request Timeout”	disconnect
409	“Conflict”	disconnect
410	“Gone”	drop
411	“Length Required”	disconnect
412	“Precondition Failed”	disconnect
413	“Payload Too Large”	disconnect
414	“URI Too Long”	disconnect

415	“Unsupported Media Type”	disconnect
416	“Range Not Satisfiable”	drop
417	“Expectation Failed”	disconnect
418	“I’m a teapot”	disconnect
421	“Misdirected Request”	disconnect
422	“Unprocessable Entity”	drop
423	“Locked”	disconnect
424	“Failed Dependency”	drop
425	“Too Early”	drop
426	“Upgrade Required”	disconnect
428	“Precondition Required”	retry
429	“Too Many Requests”	disconnect
431	“Request Header Fields Too Large”	disconnect
451	“Unavailable For Legal Reasons”	drop
500	“Internal Server Error”	disconnect
501	“Not Implemented”	disconnect
502	“Bad Gateway”	disconnect
503	“Service Unavailable”	disconnect
504	“Gateway Timeout”	retry
505	“HTTP Version Not Supported”	disconnect
506	“Variant Also Negotiates”	disconnect
507	“Insufficient Storage”	disconnect
508	“Loop Detected”	drop
510	“Not Extended”	disconnect
511	“Network Authentication Required”	disconnect

- **Message format**: Specifies the message format and normalization. Possible values:
  - **Elastic Common Schema (ECS)**: Normalize the data using the ECS schema.
  - **Open Cybersecurity Schema Framework (OCSF)**: Normalize the data using the OCSF schema.
  - **Parsed**: Send the parsed data without further normalization.

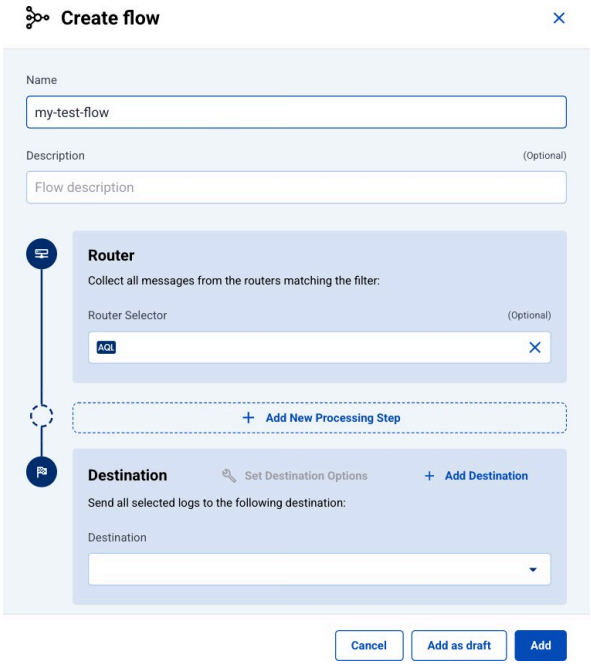
3. **Create a flow to connect the new destination to an AxoRouter instance.**

   1. Select **Flows**.
   2. Select **Add Flow > Flow**.

      To create a fallback flow, select **Add Flow > Fallback flow**.

   3. Enter a name for the flow, for example, `my-test-flow`.

4. In the **Router Selector** field, enter an expression that matches the routers you want to apply the flow to. To select a specific router, use a name selector, for example, `name = my-axorouter-hostname`.

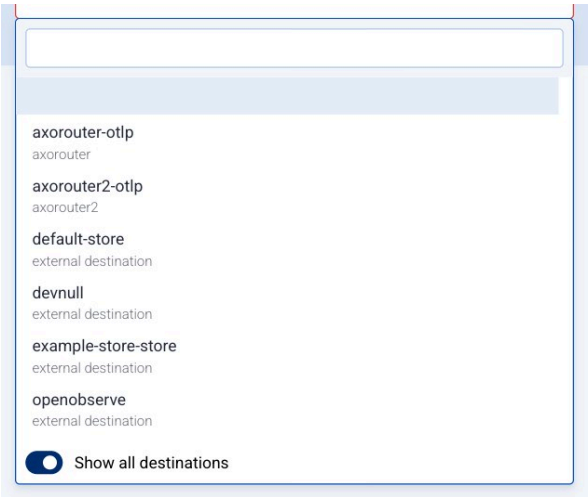
   You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

   - If you leave the router selector empty, it matches every AxoRouter instance.
   - To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.
   - If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

   > **Note**
   >
   > **You can configure multiple fallback flows, but only one fallback flow can apply to an AxoRouter (so the Router Selector of the fallback flows can't overlap).**

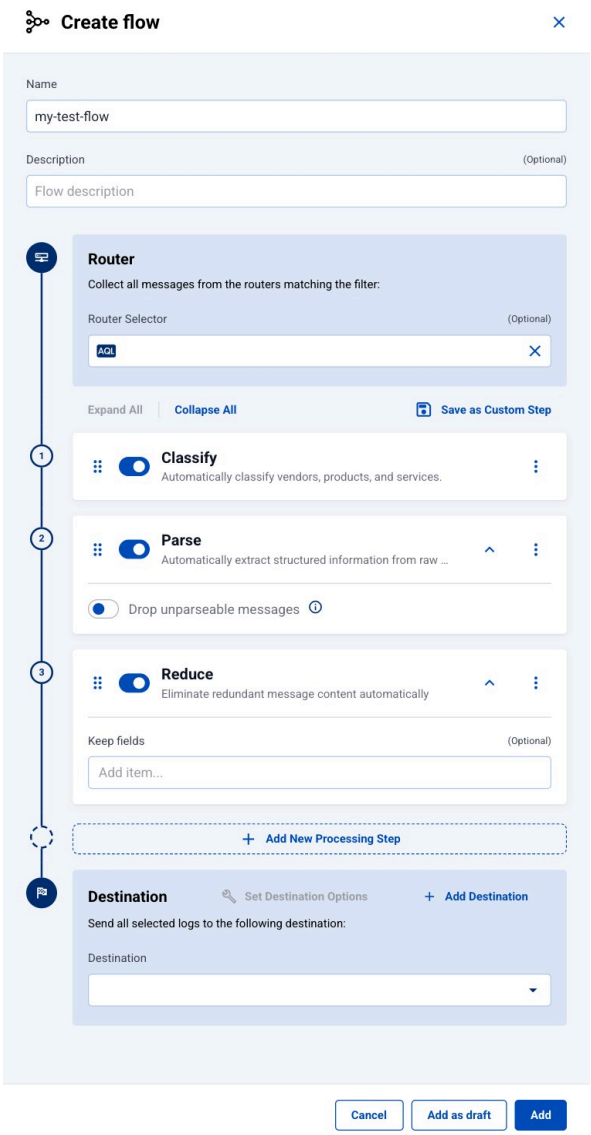
5. Select the **Destination** where you want to send your data. If you don't have any destination configured, you can select **Add Destination** in the destination section to create a new destination now. For details on the different destinations, see Destinations.

   - If you don't have any destination configured, see Destinations.
   - If you've already created a store, it's automatically available as a destination. Note that the **Router Selector** of the flow must match only AxoRouters that have the selected store available, otherwise you'll get an error message.
   - If you want to send data to another AxoRouter, enable the **Show all destinations** option, and select the connector of the AxoRouter where you want to send the data.



6. (Optional) To process the data transferred in the flow, select **Add New Processing Step**. For details, see Processing steps. For example:

    1. Add a **Classify**, a **Parse**, and a **Reduce** step, in that order, to automatically remove redundant and empty fields from your data.
    2. To select which messages are processed by the flow, add a **Select Messages** step, and enter a filter into the **AQL Expression** field. For example, to select only the messages received from Fortinet FortiGate firewalls, use the `meta.vendor = fortinet AND meta.product = fortigate` query.



7. Select **Add**, then confirm with **Add**. To save the flow without provisioning it to the routers yet, select **Add as draft** instead.

8. The new flow appears in the **Flows** list.



Related message fields

You can use the following message fields to modify messages sent to this destination using processing steps.

Troubleshooting

If the data doesn't arrive at the destination, see the data doesn't reach the destination. To find the errors that the destination returns, see the destination rejects the data. For other problems, see Troubleshooting Axoflow.

8.13 OpenSearch

Support for an OpenSearch destination is coming soon. Request it, or ask about the status of the integration.

OpenSearch: Open-source search and analytics suite for log ingestion, indexing, dashboards, and alerting.

Axoflow supports sending data to OpenSearch using its HTTP endpoint.

If you'd like to send data from AxoRouter to this destination, contact our support team for details.

8.14 Palo Alto Networks

Configure Axoflow to send your data to Palo Alto Networks Cortex XSIAM.

8.14.1 Cortex XSIAM

Add a Cortex XSIAM destination that sends data to its HTTP Log Collector, and look up the message fields it uses.

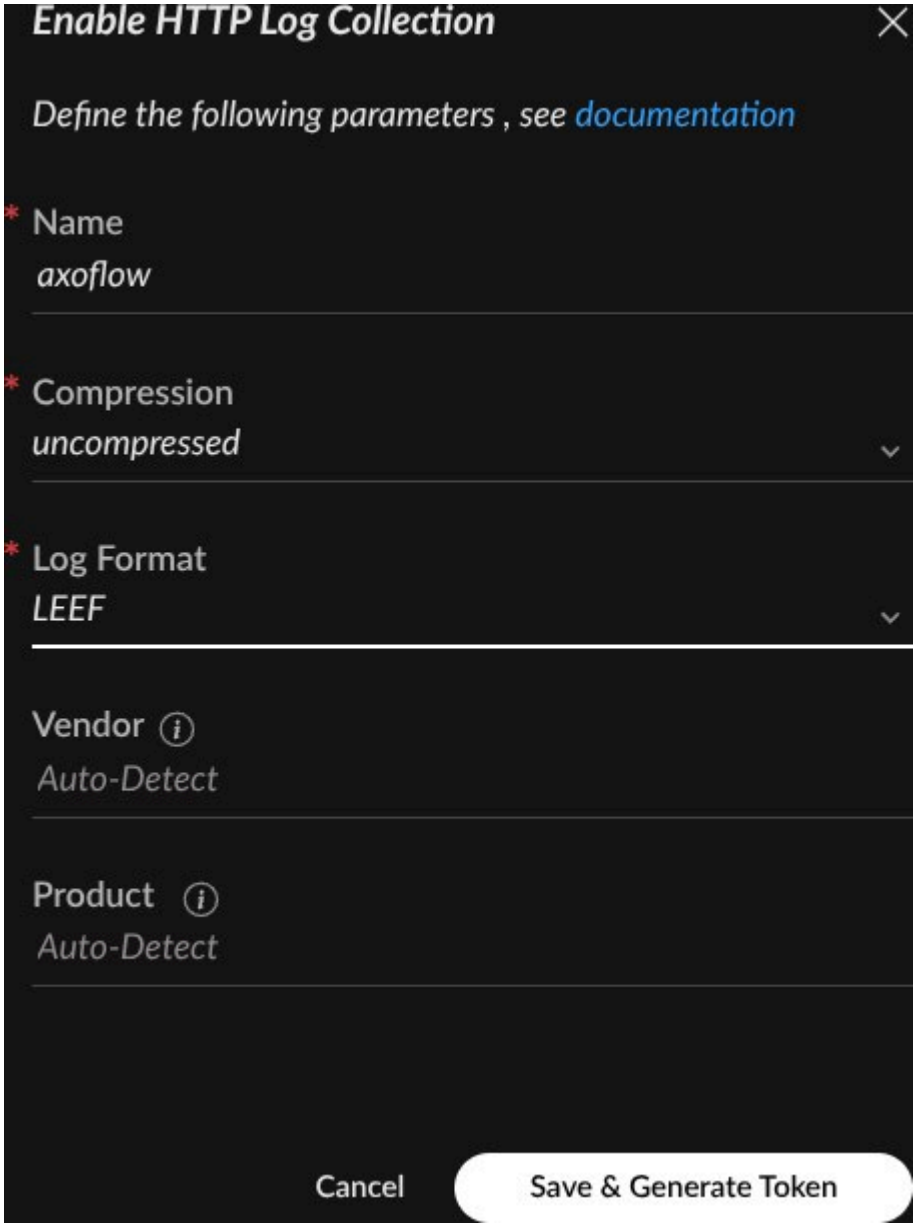
Cortex XSIAM: Extended Security Intelligence and Analytics Management (XSIAM) platform for advanced threat detection and response.

Axoflow supports sending data to Cortex XSIAM using its HTTP Log Collector.

To add an XSIAM destination to Axoflow, complete the following steps.

Prerequisites

- Set up an HTTP log collector in XSIAM.
    - Set **Log Format** to **LEEF**.



    - Record its access token. Note that XSIAM displays the token only once, if you don't save the token you'll have to create an new log collector.
    - Leave **Compression** on `uncompressed`. Compression will be supported in the next Axoflow release.

- To configure Axoflow, you'll need the URL of the Cortex XSIAM HTTP Log Collector and its access token.

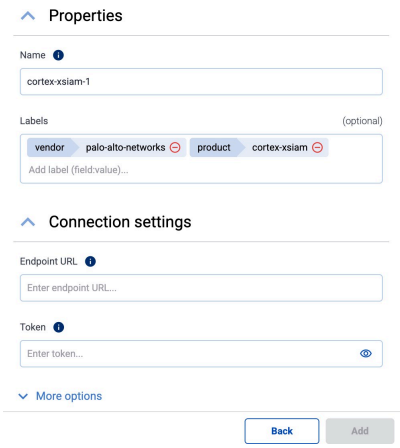
Steps

1. Create a new destination.
    1. Open the AxoConsole.
    2. Select **Destinations > Add Destination > Destination**.

2. Configure the destination.

    1. Select **Cortex XSIAM**.

    2. Enter a name for the destination.



    3. Enter the URL of the XSIAM HTTP log collector into the **Endpoint URL** field, for example: `https://api-{your-xsiam-tenant-URL}/logs/v1/event`.

    4. Enter the authentication token (api_key) you got when you created the Cortex XSIAM HTTP log collector into the **Token** field.

    5. (Optional) Set other options as needed for your environments.

        - **Timeout**: The number of seconds to wait for a log-forwarding request to complete, and attempt to reconnect the server if exceeded. If the timeout is exceeded, AxoRouter attempts to reconnect the destination. The default (`0`) is unlimited.

          For more details, see the AxoSyslog documentation.

        - **Disk buffer**: Explicitly enable or disable disk buffer on AxoRouter. Default value: **Enabled**.

          AxoRouter can store messages on the local disk if the destination or the network connection to the destination becomes unavailable. AxoRouter automatically sends the stored messages (in the order they were received) to the destination when the connection is reestablished. Note that the disk buffer is separate from AxoStore.

          Disk buffer capacity (per worker): Size of the disk buffer files in bytes. This value applies for each worker. Default value is 100MiB (104857600 bytes), which is also the recommended minimum.

          The disk required for using disk buffering is **Disk buffer capacity (per worker)** * **Number of Workers** for each destination of AxoRouter.

- **Batch Size**: Sets the maximum size of payload in a batch. If the size of the messages reaches this value, AxoRouter sends the batch to the destination even if the number of messages is less than the value of the batch lines option.

  > **CAUTION:**
  >
  > When setting **Batch Size**:
  >
  > - Note that the actual HTTP request will be about 4KB larger because of the request headers.
  > - Check the configuration of your destination, as SaaS solutions and ingress controllers often limit the maximal size of HTTP requests. If the sum of **Batch Size** and the request headers exceeds this limit, the data will be rejected. (In case of NGINX, with a `client intended to send too large body` error message.)

- **Batch Lines**: Number of lines sent to the destination in one batch. AxoRouter waits for this number of lines to accumulate and sends them off in a single batch. Increasing this number increases throughput as more messages are sent in a single batch, but also increases message latency.

  Note that **Initial window size** * **Maximum Connections** must be greater or equal than **Batch lines** * **Number of workers** for all the sources that feeds a given destination. Otherwise, the destination worker might be unable to send data until **Batch timeout** expires, even if there's data available.

  For more details, see the AxoSyslog documentation.

- **Batch Timeout**: Maximal time in milliseconds to wait for a batch to be filled before sending it. The default value (`-1`) means that the batch should be full before sending, which may result in a long wait if the incoming traffic is low.

  For more details, see the AxoSyslog documentation.

- **Number of Workers**: Used for scaling the destination in case of high message load. Specifies the number of worker threads AxoRouter uses for sending messages to the destination. The default is 1. If high message throughput is not a concern, leave it on 1. For maximum performance, increase it up to the number of CPU cores available on AxoRouter.

  For more details, see the AxoSyslog documentation.

- **Message format**: Specifies the message format and normalization. Possible values:

- **Cortex Data Model (XDM)**: Send the normalized data to XSIAM. The Axoflow integration in XSIAM puts the data to its proper place, without having to use XSIAM parsers. Contact our support team for details.
- **Dataset**: Send the processed data in a format that XSIAM parsers can process.
- **Text**: Send the raw data in LEEF format.
- **Elastic Common Schema (ECS)**: Normalize the data using the ECS schema.
- **Open Cybersecurity Schema Framework (OCSF)**: Normalize the data using the OCSF schema.
- **Parsed**: Send the parsed data without further normalization.

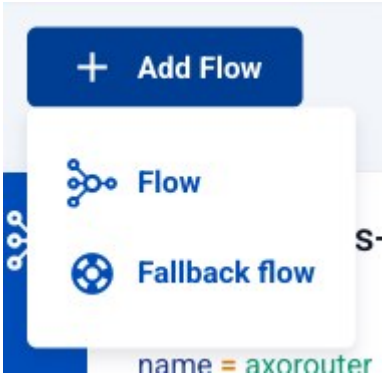
6. Select **Add**.

3. **Create a flow to connect the new destination to an AxoRouter instance.**

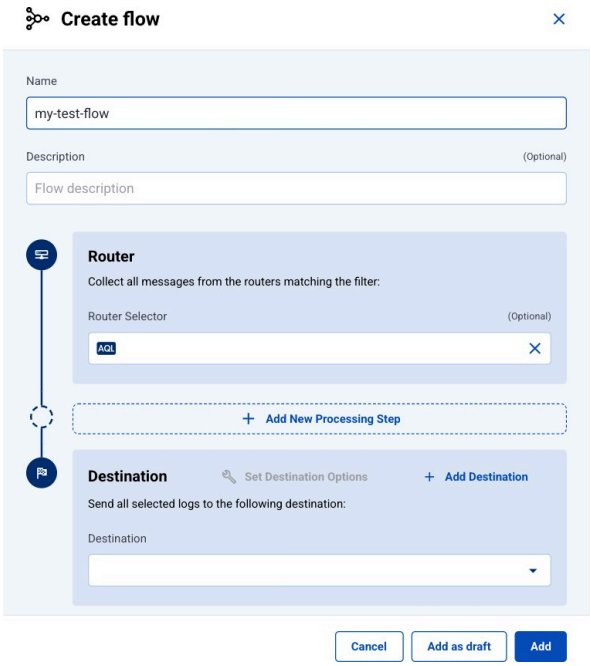
   1. Select **Flows**.

   2. Select **Add Flow > Flow**.

      To create a fallback flow, select **Add Flow > Fallback flow**.

   3. Enter a name for the flow, for example, `my-test-flow`.



4. In the **Router Selector** field, enter an expression that matches the routers you want to apply the flow to. To select a specific router, use a name selector, for example, `name = my-axorouter-hostname`.

   You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

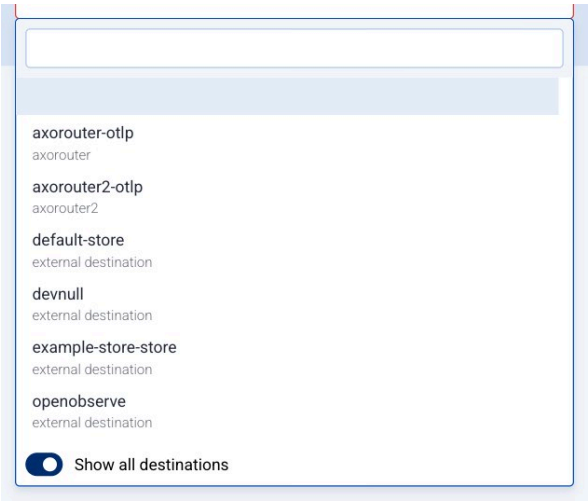
   - If you leave the router selector empty, it matches every AxoRouter instance.
   - To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.
   - If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

   > **Note**
   >
   > **You can configure multiple fallback flows, but only one fallback flow can apply to an AxoRouter (so the Router Selector of the fallback flows can't overlap).**

5. Select the **Destination** where you want to send your data. If you don't have any destination configured, you can select **Add Destination** in the destination section to create a new destination now. For details on the different destinations, see Destinations.

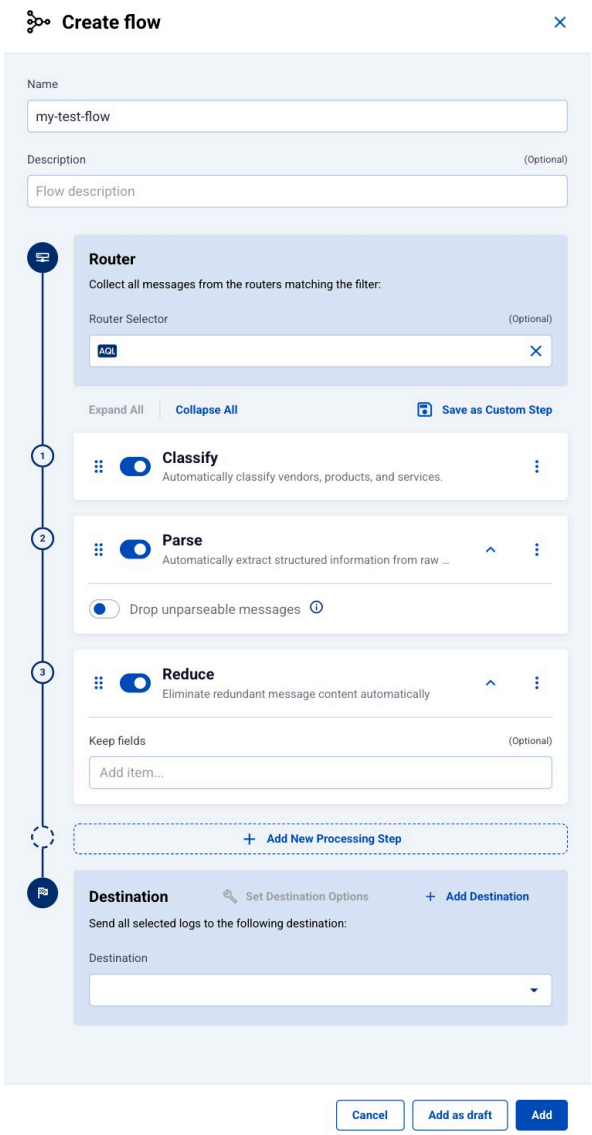
   - If you don't have any destination configured, see Destinations.

- If you've already created a store, it's automatically available as a destination. Note that the **Router Selector** of the flow must match only AxoRouters that have the selected store available, otherwise you'll get an error message.
- If you want to send data to another AxoRouter, enable the **Show all destinations** option, and select the connector of the AxoRouter where you want to send the data.



6. (Optional) To process the data transferred in the flow, select **Add New Processing Step**. For details, see Processing steps. For example:

   1. Add a **Classify**, a **Parse**, and a **Reduce** step, in that order, to automatically remove redundant and empty fields from your data.
   2. To select which messages are processed by the flow, add a **Select Messages** step, and enter a filter into the **AQL Expression** field. For example, to select only the messages received from Fortinet FortiGate firewalls, use the `meta.vendor = fortinet AND meta.product = fortigate` query.



7. Select **Add**, then confirm with **Add**. To save the flow without provisioning it to the routers yet, select **Add as draft** instead.

8. The new flow appears in the **Flows** list.



Related message fields

You can use the following message fields to modify messages sent to this destination using processing steps.

Troubleshooting

If the data doesn't arrive at the destination, see the data doesn't reach the destination. To find the errors that the destination returns, see the destination rejects the data. For other problems, see Troubleshooting Axoflow.

8.15 Splunk

Add a Splunk Cloud or Splunk Enterprise destination, and look up the message fields that modify the data sent to Splunk.

Splunk: Ingests, indexes, and visualizes data for monitoring, analysis, and alerting.

To add a Splunk destination (Splunk Cloud or Splunk Enterprise) to Axoflow, complete the following steps.

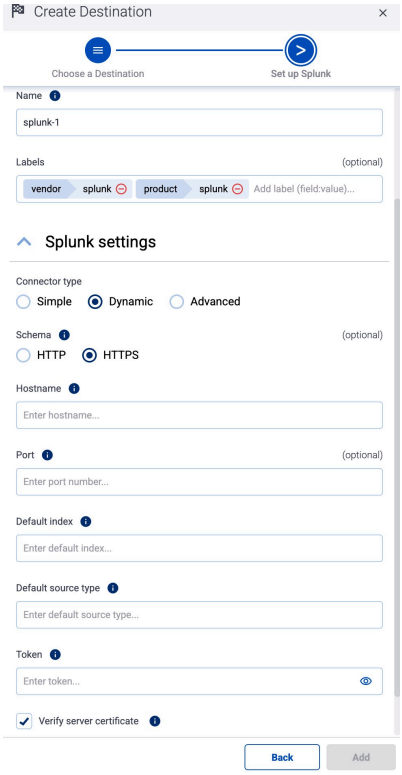
Prerequisites

1. Enable the HTTP Event Collector (HEC) on your Splunk deployment if needed. On Splunk Cloud Platform deployments, HEC is enabled by default.
2. Create a token for Axoflow to use in the destination. When creating the token, use the syslog source type.

   For details, see Set up and use HTTP Event Collector in Splunk Web.
3. If you're using AxoRouter, create the indexes where Axoflow sends the log data. Which index is needed depends on the sources you have, but create at least the following event indices: `axoflow`, `infraops`, `netops`, `netfw`, `osnix` (for unclassified messages). Check your sources in the Sources section for a detailed lists on which indices their data is sent.
4. If you've created any new indexes, make sure to add those indexes to the token's Allowed Indexes.

Steps

1. Create a new destination.
   1. Open the AxoConsole.
   2. Select **Destinations > Add Destination > Destination**.
2. Configure the destination.
   1. Select **Splunk**.
   2. Enter a name for the destination.



3. Select which type of configuration you want to use:
   - **Simple**: Send all data into a single index, with fixed source and source type settings.
   - **Dynamic**: Set index, source, and source type based on the content or metadata of the incoming messages.
   - **Advanced**: Allows you to specify a custom URL endpoint.
4. Configure the endpoint of the destination.
   - **Advanced**: Enter your Splunk URL into the **URL** field, for example, `https://<your-splunk-tenant-id>.splunkcloud.com:8088` for Splunk Cloud Platform free trials, or `https://<your-splunk-tenant-id>.splunkcloud.com` for Splunk Cloud Platform instances.
   - **Simple** and **Dynamic**:
     1. Select the **HTTPS** or **HTTP** protocol to use to access your destination.
     2. Enter the **Hostname** and **Port** of the destination.
5. Specify the Splunk index to send the data to.
   - **Simple**: Enter the expression that specifies the Splunk index to use into the **Index** field, for example: `netops`. All data will be sent into this index.
   - **Dynamic** and **Advanced**:
     1. Enter the name of the **Default Index**. The data will be sent into this index if no other index is set during the processing of the message (based on automatic classification, or by the processing steps of the **Flow**). Make sure that the index exists in Splunk.

2. Enter the **Default Source Type**. This will be assigned to the messages that have no sourcetype set during the processing of the message (based on automatic classification, or by the processing steps of the **Flow**).

6. Enter the token you've created into the **Token** field.

7. Disable the **Verify server certificate** option unless your deployment has a valid, non-self-signed certificate. Free Splunk Cloud accounts have self-signed certificates.

8. (Optional) Set other options as needed for your environments.

   - **Timeout**: The number of seconds to wait for a log-forwarding request to complete, and attempt to reconnect the server if exceeded. If the timeout is exceeded, AxoRouter attempts to reconnect the destination. The default (`0`) is unlimited.

     For more details, see the AxoSyslog documentation.

   - **Disk buffer**: Explicitly enable or disable disk buffer on AxoRouter. Default value: **Enabled**.

     AxoRouter can store messages on the local disk if the destination or the network connection to the destination becomes unavailable. AxoRouter automatically sends the stored messages (in the order they were received) to the destination when the connection is reestablished. Note that the disk buffer is separate from AxoStore.

     Disk buffer capacity (per worker): Size of the disk buffer files in bytes. This value applies for each worker. Default value is 100MiB (104857600 bytes), which is also the recommended minimum.

     The disk required for using disk buffering is **Disk buffer capacity (per worker)** * **Number of Workers** for each destination of AxoRouter.

   - **Batch Size**: Sets the maximum size of payload in a batch. If the size of the messages reaches this value, AxoRouter sends the batch to the destination even if the number of messages is less than the value of the batch lines option.

     > **CAUTION:**
     >
     > When setting **Batch Size**:
     >
     > - Note that the actual HTTP request will be about 4KB larger because of the request headers.

- Check the configuration of your destination, as SaaS solutions and ingress controllers often limit the maximal size of HTTP requests. If the sum of **Batch Size** and the request headers exceeds this limit, the data will be rejected. (In case of NGINX, with a `client intended to send too large body` error message.)

- **Batch Lines**: Number of lines sent to the destination in one batch. AxoRouter waits for this number of lines to accumulate and sends them off in a single batch. Increasing this number increases throughput as more messages are sent in a single batch, but also increases message latency.

  Note that **Initial window size** * **Maximum Connections** must be greater or equal than **Batch lines** * **Number of workers** for all the sources that feeds a given destination. Otherwise, the destination worker might be unable to send data until **Batch timeout** expires, even if there's data available.

  For more details, see the AxoSyslog documentation.

- **Batch Timeout**: Maximal time in milliseconds to wait for a batch to be filled before sending it. The default value (`-1`) means that the batch should be full before sending, which may result in a long wait if the incoming traffic is low.

  For more details, see the AxoSyslog documentation.

- **Content compression**: Specifies the compression to use when sending data to the destination. `No compression` explicitly disables compression. Other possible values: `Gzip` , `Deflate` . Select only compression that your destination supports and has enabled. Default value: `No compression`

- **Number of retries**: The number of retry attempts when the retry action is triggered. Default: `3` .

  For more details, see the AxoSyslog documentation.

- **Time to reopen**: The time to wait before reconnecting when a connection fails. Default: `1` second.

  For more details, see the AxoSyslog documentation.

- **Timeout**: The number of seconds to wait for a log-forwarding request to complete, and attempt to reconnect the server if exceeded. If the timeout is exceeded, AxoRouter attempts to reconnect the destination. The default (`0`) is unlimited.

  For more details, see the AxoSyslog documentation.

- **Number of Workers**: Used for scaling the destination in case of high message load. Specifies the number of worker threads AxoRouter uses for sending messages to the destination. The default is 1. If high message throughput is not a concern, leave it on 1. For maximum performance, increase it up to the number of CPU cores available on AxoRouter.

  For more details, see the AxoSyslog documentation.

- **Response actions**: Specifies what AxoRouter does with the log message, based on the response code received from the destination.

  - If the status code begins with 2 (for example, `200`), AxoRouter assumes the message was successfully sent.
  - If you've configured a custom action in AxoConsole for the status code, AxoRouter performs the specified action.
  - If the status code appears in the default response actions table, AxoRouter performs the specified action.
  - For other status codes, AxoRouter disconnects.

  To add a custom response action, select **Add new action**, then enter the **HTTP status code** and the related **Action**. The following actions are available:

  - **Disconnect**: Keep trying to resend the message indefinitely.
  - **Drop**: Drop the message without trying to resend it.
  - **Retry**: Retry sending the message for a maximum of 3 times.
  - **Success**: Assume the message was successfully sent.

> **Show default actions**
>
> | code | explanation | action |
> |---|---|---|
> | 100 | "Continue" | disconnect |
> | 101 | "Switching Protocols" | disconnect |
> | 102 | "Processing" | retry |
> | 103 | "Early Hints" | retry |
> | 200 | "OK" | success |
> | 201 | "Created" | success |
> | 202 | "Accepted" | success |
> | 203 | "Non-Authoritative Information" | success |

204	“No Content”	success
205	“Reset Content”	success
206	“Partial Content”	success
300	“Multiple Choices”	disconnect
301	“Moved Permanently”	disconnect
302	“Found”	disconnect
303	“See Other”	disconnect
304	“Not Modified”	retry
307	“Temporary Redirect”	disconnect
308	“Permanent Redirect”	disconnect
400	“Bad Request”	disconnect
401	“Unauthorized”	disconnect
402	“Payment Required”	disconnect
403	“Forbidden”	disconnect
404	“Not Found”	disconnect
405	“Method Not Allowed”	disconnect
406	“Not Acceptable”	disconnect
407	“Proxy Authentication Required”	disconnect
408	“Request Timeout”	disconnect
409	“Conflict”	disconnect
410	“Gone”	drop
411	“Length Required”	disconnect
412	“Precondition Failed”	disconnect
413	“Payload Too Large”	disconnect
414	“URI Too Long”	disconnect

415	“Unsupported Media Type”	disconnect
416	“Range Not Satisfiable”	drop
417	“Expectation Failed”	disconnect
418	“I’m a teapot”	disconnect
421	“Misdirected Request”	disconnect
422	“Unprocessable Entity”	drop
423	“Locked”	disconnect
424	“Failed Dependency”	drop
425	“Too Early”	drop
426	“Upgrade Required”	disconnect
428	“Precondition Required”	retry
429	“Too Many Requests”	disconnect
431	“Request Header Fields Too Large”	disconnect
451	“Unavailable For Legal Reasons”	drop
500	“Internal Server Error”	disconnect
501	“Not Implemented”	disconnect
502	“Bad Gateway”	disconnect
503	“Service Unavailable”	disconnect
504	“Gateway Timeout”	retry
505	“HTTP Version Not Supported”	disconnect
506	“Variant Also Negotiates”	disconnect
507	“Insufficient Storage”	disconnect
508	“Loop Detected”	drop
510	“Not Extended”	disconnect
511	“Network Authentication Required”	disconnect

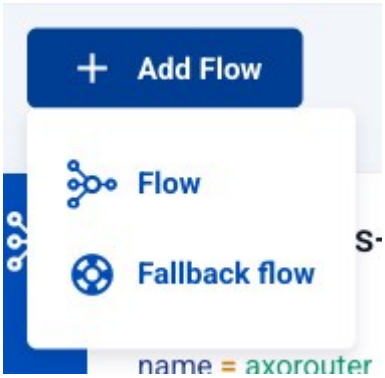
9. Select **Add**.

3. **Create a flow to connect the new destination to an AxoRouter instance.**

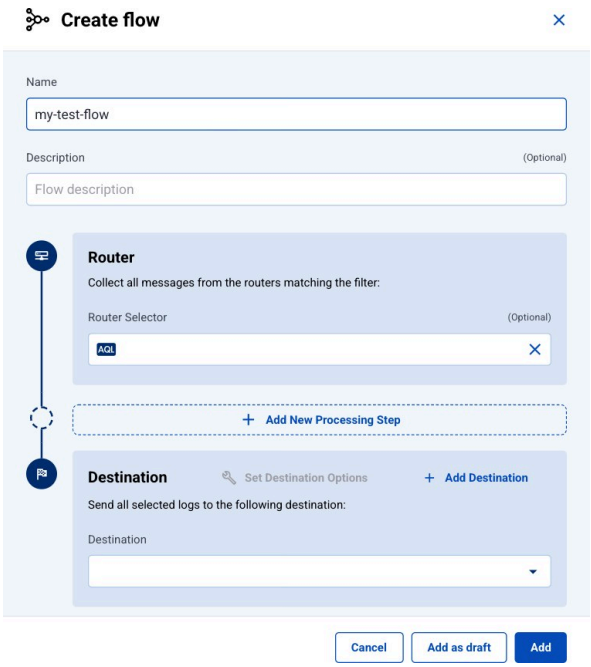
   1. Select **Flows**.

   2. Select **Add Flow > Flow**.

      To create a fallback flow, select **Add Flow > Fallback flow**.

   3. Enter a name for the flow, for example, `my-test-flow`.

   4. In the **Router Selector** field, enter an expression that matches the routers you want to apply the flow to. To select a specific router, use a name selector, for example, `name = my-axorouter-hostname`.

      You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

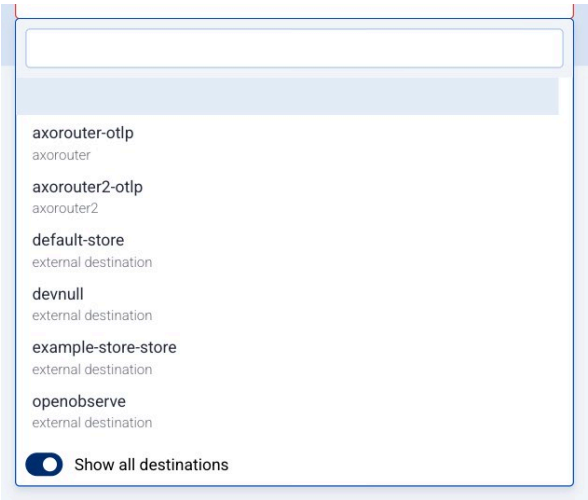
      - If you leave the router selector empty, it matches every AxoRouter instance.
      - To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.

- If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

> **Note**
>
> **You can configure multiple fallback flows, but only one fallback flow can apply to an AxoRouter (so the Router Selector of the fallback flows can't overlap).**

5. Select the **Destination** where you want to send your data. If you don't have any destination configured, you can select **Add Destination** in the destination section to create a new destination now. For details on the different destinations, see Destinations.

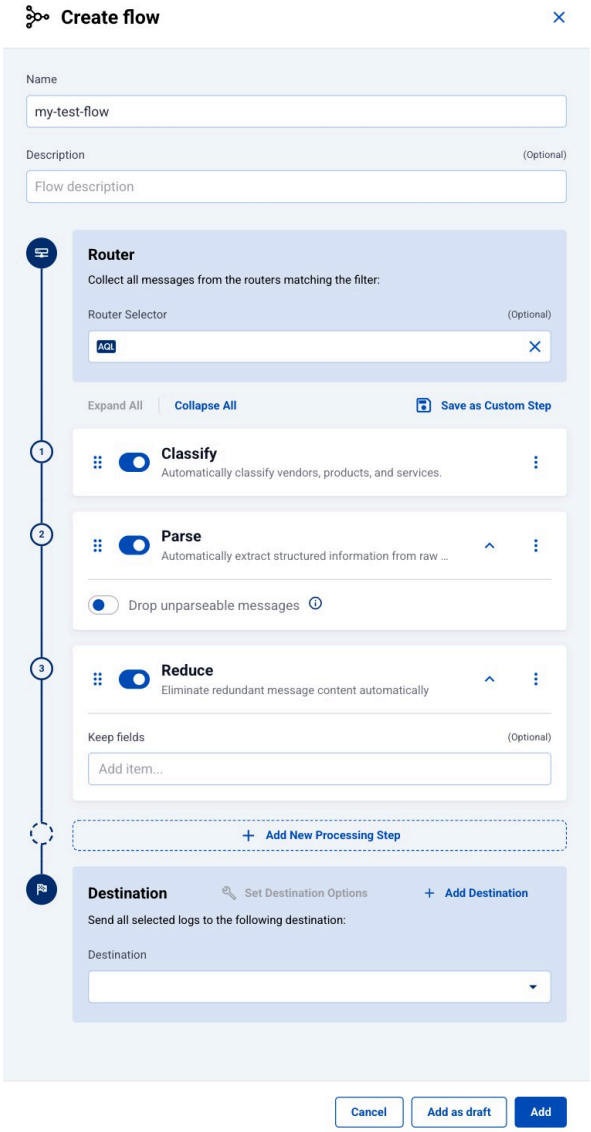
   - If you don't have any destination configured, see Destinations.
   - If you've already created a store, it's automatically available as a destination. Note that the **Router Selector** of the flow must match only AxoRouters that have the selected store available, otherwise you'll get an error message.
   - If you want to send data to another AxoRouter, enable the **Show all destinations** option, and select the connector of the AxoRouter where you want to send the data.



6. (Optional) To process the data transferred in the flow, select **Add New Processing Step**. For details, see Processing steps. For example:

   1. Add a **Classify**, a **Parse**, and a **Reduce** step, in that order, to automatically remove redundant and empty fields from your data.
   2. To select which messages are processed by the flow, add a **Select Messages** step, and enter a filter into the **AQL Expression** field. For example, to select only the messages received from Fortinet FortiGate

firewalls, use the `meta.vendor = fortinet AND meta.product = fortigate` query.



7. Select **Add**, then confirm with **Add**. To save the flow without provisioning it to the routers yet, select **Add as draft** instead.

8. The new flow appears in the **Flows** list.



Related message fields

You can use the following message fields to modify messages sent to this destination using processing steps.

Troubleshooting

If the data doesn't arrive at the destination, see the data doesn't reach the destination. To find the errors that the destination returns, see the destination rejects the data. For other problems, see Troubleshooting Axoflow.

8.16 Sumo Logic

Add a Sumo Logic destination to send your security data to Sumo Logic, and look up the message fields it uses.

Sumo Logic: Cloud-native platform for log management, metrics analysis, and security analytics.

To add a Sumo Logic destination to Axoflow, complete the following steps.

Prerequisites

- A Sumo Logic account.
- A HTTP Hosted Collector configured in the Sumo Logic service.

  To configure a Hosted Collector, follow the configuration instructions under the Configure a Hosted Collector section on the official Sumo Logic website.
- The unique HTTP collector code you receive while configuring your Host Collector for HTTP requests.
- To add custom fields of the messages to Sumo Logic:

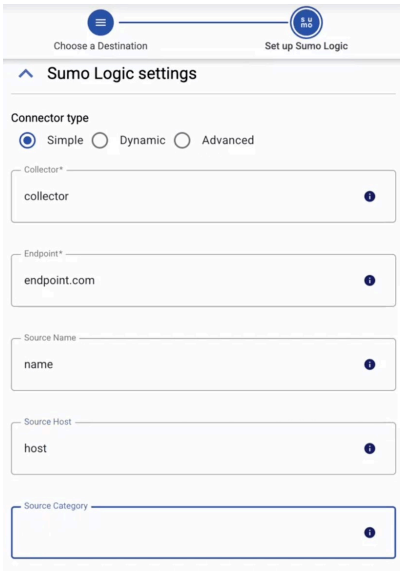
  1. These fields must already exists in your Sumo Logic account. For details, see the Sumo Logic documentation.
  2. You must configure the HTTP Hosted Collector to process the received fields.

Steps

1. Create a new destination.

   1. Open the AxoConsole.
   2. Select **Destinations > Add Destination > Destination**.
2. Configure the destination.

   1. Select **Sumo Logic**.
   2. Enter a name for the destination.



3. Select which type of configuration you want to use:
   - **Simple**: Send all data using a single source host and source category.
   - **Dynamic**: Set source host, name, and category based on the content or metadata of the incoming messages.
4. Configure the endpoint of the destination.
   - **Simple**:
     1. Enter the **URL** of the HTTPS endpoint of your Sumo Logic HTTP Source where data is sent. (AxoConsole UI doesn't show the URL by default because it contains an access token.)
     2. Enter the name of the **Source Name**. All data will be associated as coming from this source.
     3. Enter the host identifier (for example, server or container name) that sent the data into the **Source Host** field. All data will be associated as coming from this source.
     4. Enter the name of the **Source Category**. The data will be added to this category in Sumo Logic.
     5. Enter the metadata tags that to be assigned to your data by default into the **Fields** field. Use key-value pairs. Other metadata can be added during the processing of the message (based on automatic classification, or by the processing steps of the **Flow**). Note the following limits of Sumo Logic fields:
        - An HTTP request is limited to 30 fields.
        - A field name (key) cannot be longer than 255 characters.
        - A value cannot be longer than 200 characters.
        - You must configure your Sumo Logic instance to accept and process the received fields.

- **Dynamic**:

  1. Enter the **URL** of the HTTPS endpoint of your Sumo Logic HTTP Source where data is sent. (AxoConsole UI doesn't show the URL by default because it contains an access token.)
  2. Enter the **Default Source Name**. The name of the data source. The data will be associated as coming from this source, unless a different one is set during the processing of the message (based on automatic classification, or by the processing steps of the **Flow**).
  3. Enter the **Default Source Host**. The host identifier (for example, server or container name) that sent the data. All data will be associated as coming from this source, unless a different one is set during the processing of the message (based on automatic classification, or by the processing steps of the **Flow**).
  4. Enter the name of the **Default Source Category**. This is a user-defined tag to logically group or filter data in Sumo Logic. The data will added to this category, unless a different one is set during the processing of the message (based on automatic classification, or by the processing steps of the **Flow**).
  5. Enter the metadata tags that to be assigned to your data by default into the **Fields** field. Use key-value pairs. Other metadata can be added during the processing of the message (based on automatic classification, or by the processing steps of the **Flow**). Note the following limits of Sumo Logic fields:

     - An HTTP request is limited to 30 fields.
     - A field name (key) cannot be longer than 255 characters.
     - A value cannot be longer than 200 characters.
     - You must configure your Sumo Logic instance to accept and process the received fields.

5. (Optional) Set other options as needed for your environments.

   - **Timeout**: The number of seconds to wait for a log-forwarding request to complete, and attempt to reconnect the server if exceeded. If the timeout is exceeded, AxoRouter attempts to reconnect the destination. The default (`0`) is unlimited.

     For more details, see the AxoSyslog documentation.

   - **Disk buffer**: Explicitly enable or disable disk buffer on AxoRouter. Default value: **Enabled**.

AxoRouter can store messages on the local disk if the destination or the network connection to the destination becomes unavailable. AxoRouter automatically sends the stored messages (in the order they were received) to the destination when the connection is reestablished. Note that the disk buffer is separate from AxoStore.

Disk buffer capacity (per worker): Size of the disk buffer files in bytes. This value applies for each worker. Default value is 100MiB (104857600 bytes), which is also the recommended minimum.

The disk required for using disk buffering is **Disk buffer capacity (per worker)** * **Number of Workers** for each destination of AxoRouter.

- **Batch Size**: Sets the maximum size of payload in a batch. If the size of the messages reaches this value, AxoRouter sends the batch to the destination even if the number of messages is less than the value of the batch lines option.

  > **CAUTION:**
  >
  > When setting **Batch Size**:
  >
  > - Note that the actual HTTP request will be about 4KB larger because of the request headers.
  > - Check the configuration of your destination, as SaaS solutions and ingress controllers often limit the maximal size of HTTP requests. If the sum of **Batch Size** and the request headers exceeds this limit, the data will be rejected. (In case of NGINX, with a `client intended to send too large body` error message.)

- **Batch Lines**: Number of lines sent to the destination in one batch. AxoRouter waits for this number of lines to accumulate and sends them off in a single batch. Increasing this number increases throughput as more messages are sent in a single batch, but also increases message latency.

  Note that **Initial window size** * **Maximum Connections** must be greater or equal than **Batch lines** * **Number of workers** for all the sources that feeds a given destination. Otherwise, the destination worker might be unable to send data until **Batch timeout** expires, even if there's data available.

  For more details, see the AxoSyslog documentation.

- **Batch Timeout**: Maximal time in milliseconds to wait for a batch to be filled before sending it. The default value (`-1`) means that the batch should be full before sending, which may result in a long wait if the incoming traffic is low.

  For more details, see the AxoSyslog documentation.

- **Content compression**: Specifies the compression to use when sending data to the destination. `No compression` explicitly disables compression. Other possible values: `Gzip`, `Deflate`. Select only compression that your destination supports and has enabled. Default value: `No compression`
- **Number of retries**: The number of retry attempts when the retry action is triggered. Default: `3`.

  For more details, see the AxoSyslog documentation.
- **Time to reopen**: The time to wait before reconnecting when a connection fails. Default: `1` second.

  For more details, see the AxoSyslog documentation.
- **Timeout**: The number of seconds to wait for a log-forwarding request to complete, and attempt to reconnect the server if exceeded. If the timeout is exceeded, AxoRouter attempts to reconnect the destination. The default (`0`) is unlimited.

  For more details, see the AxoSyslog documentation.
- **Number of Workers**: Used for scaling the destination in case of high message load. Specifies the number of worker threads AxoRouter uses for sending messages to the destination. The default is 1. If high message throughput is not a concern, leave it on 1. For maximum performance, increase it up to the number of CPU cores available on AxoRouter.

  For more details, see the AxoSyslog documentation.
- **Response actions**: Specifies what AxoRouter does with the log message, based on the response code received from the destination.
  - If the status code begins with 2 (for example, `200`), AxoRouter assumes the message was successfully sent.
  - If you've configured a custom action in AxoConsole for the status code, AxoRouter performs the specified action.
  - If the status code appears in the default response actions table, AxoRouter performs the specified action.
  - For other status codes, AxoRouter disconnects.

  To add a custom response action, select **Add new action**, then enter the **HTTP status code** and the related **Action**. The following actions are available:
  - **Disconnect**: Keep trying to resend the message indefinitely.
  - **Drop**: Drop the message without trying to resend it.
  - **Retry**: Retry sending the message for a maximum of 3 times.

- **Success**: Assume the message was successfully sent.

Show default actions

code	explanation	action
100	“Continue”	disconnect
101	“Switching Protocols”	disconnect
102	“Processing”	retry
103	“Early Hints”	retry
200	“OK”	success
201	“Created”	success
202	“Accepted”	success
203	“Non-Authoritative Information”	success
204	“No Content”	success
205	“Reset Content”	success
206	“Partial Content”	success
300	“Multiple Choices”	disconnect
301	“Moved Permanently”	disconnect
302	“Found”	disconnect
303	“See Other”	disconnect
304	“Not Modified”	retry
307	“Temporary Redirect”	disconnect
308	“Permanent Redirect”	disconnect
400	“Bad Request”	disconnect
401	“Unauthorized”	disconnect
402	“Payment Required”	disconnect
403	“Forbidden”	disconnect
404	“Not Found”	disconnect

405	“Method Not Allowed”	disconnect
406	“Not Acceptable”	disconnect
407	“Proxy Authentication Required”	disconnect
408	“Request Timeout”	disconnect
409	“Conflict”	disconnect
410	“Gone”	drop
411	“Length Required”	disconnect
412	“Precondition Failed”	disconnect
413	“Payload Too Large”	disconnect
414	“URI Too Long”	disconnect
415	“Unsupported Media Type”	disconnect
416	“Range Not Satisfiable”	drop
417	“Expectation Failed”	disconnect
418	“I’m a teapot”	disconnect
421	“Misdirected Request”	disconnect
422	“Unprocessable Entity”	drop
423	“Locked”	disconnect
424	“Failed Dependency”	drop
425	“Too Early”	drop
426	“Upgrade Required”	disconnect
428	“Precondition Required”	retry
429	“Too Many Requests”	disconnect
431	“Request Header Fields Too Large”	disconnect
451	“Unavailable For Legal Reasons”	drop
500	“Internal Server Error”	disconnect

501	“Not Implemented”	disconnect
502	“Bad Gateway”	disconnect
503	“Service Unavailable”	disconnect
504	“Gateway Timeout”	retry
505	“HTTP Version Not Supported”	disconnect
506	“Variant Also Negotiates”	disconnect
507	“Insufficient Storage”	disconnect
508	“Loop Detected”	drop
510	“Not Extended”	disconnect
511	“Network Authentication Required”	disconnect

- **Message format**: Specifies the message format and normalization. Possible values:
  - **Elastic Common Schema (ECS)**: Normalize the data using the ECS schema.
  - **Open Cybersecurity Schema Framework (OCSF)**: Normalize the data using the OCSF schema.
  - **Parsed**: Send the parsed data without further normalization.

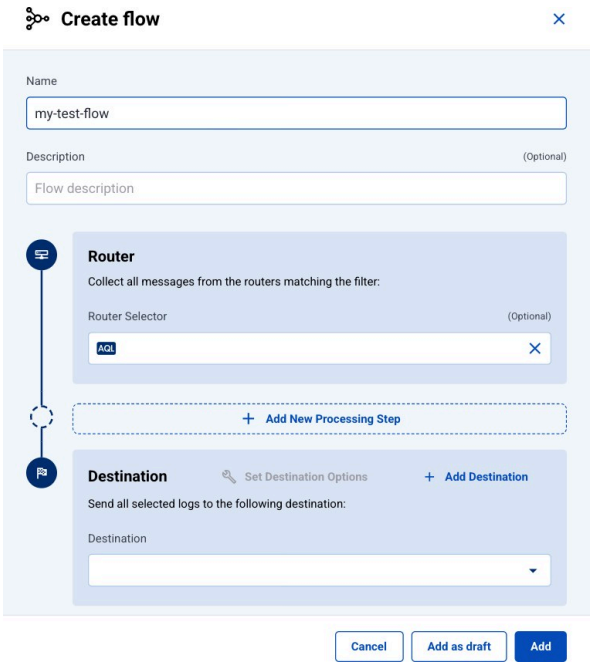
3. **Create a flow to connect the new destination to an AxoRouter instance.**

   1. Select **Flows**.
   2. Select **Add Flow > Flow**.

      To create a fallback flow, select **Add Flow > Fallback flow**.

   3. Enter a name for the flow, for example, `my-test-flow`.



4. In the **Router Selector** field, enter an expression that matches the routers you want to apply the flow to. To select a specific router, use a name selector, for example, `name = my-axorouter-hostname`.

   You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

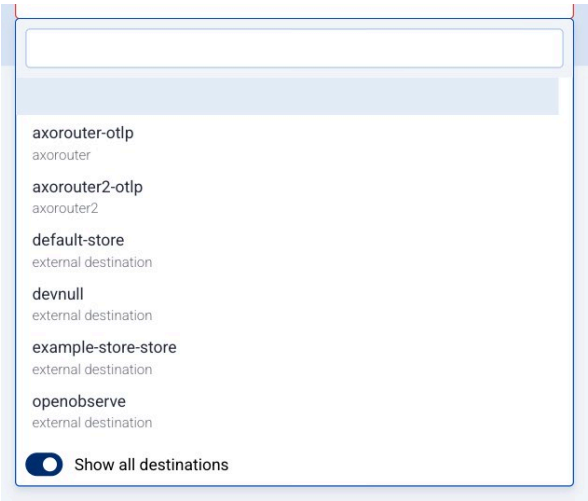
   - If you leave the router selector empty, it matches every AxoRouter instance.
   - To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.
   - If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

   > **Note**
   >
   > **You can configure multiple fallback flows, but only one fallback flow can apply to an AxoRouter (so the Router Selector of the fallback flows can't overlap).**

5. Select the **Destination** where you want to send your data. If you don't have any destination configured, you can select **Add Destination** in the destination section to create a new destination now. For details on the different destinations, see Destinations.

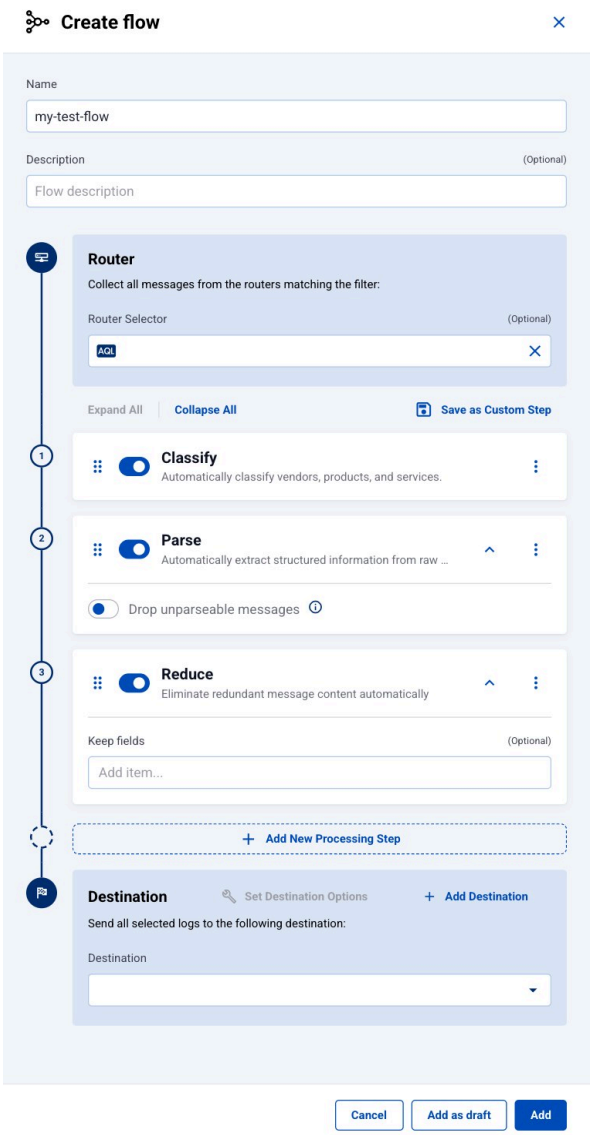
   - If you don't have any destination configured, see Destinations.

- If you've already created a store, it's automatically available as a destination. Note that the **Router Selector** of the flow must match only AxoRouters that have the selected store available, otherwise you'll get an error message.
- If you want to send data to another AxoRouter, enable the **Show all destinations** option, and select the connector of the AxoRouter where you want to send the data.



6. (Optional) To process the data transferred in the flow, select **Add New Processing Step**. For details, see Processing steps. For example:

   1. Add a **Classify**, a **Parse**, and a **Reduce** step, in that order, to automatically remove redundant and empty fields from your data.
   2. To select which messages are processed by the flow, add a **Select Messages** step, and enter a filter into the **AQL Expression** field. For example, to select only the messages received from Fortinet FortiGate firewalls, use the `meta.vendor = fortinet AND meta.product = fortigate` query.



7. Select **Add**, then confirm with **Add**. To save the flow without provisioning it to the routers yet, select **Add as draft** instead.

8. The new flow appears in the **Flows** list.



Related message fields

You can use the following message fields to modify messages sent to this destination using processing steps.

Troubleshooting

If the data doesn't arrive at the destination, see the data doesn't reach the destination. To find the errors that the destination returns, see the destination rejects the data. For other problems, see Troubleshooting Axoflow.

9 Storage

Create and manage AxoStore and external stores, search stored data, rehydrate archived logs, and check storage metrics.

Axoflow can keep a copy of your security data in *stores*, so you can search it later, or send it to a SIEM when you need it. AxoStore keeps the data locally on your AxoRouter hosts, while external stores keep it in AxoLake or in a self-managed ClickHouse instance. For details, see the storage overview.

- To create a store, see Create Store, Create External Store, or Create Logstore. A new store appears as a destination that you can select in your flows.
- To query the stored data, see Search.
- To send stored data to another destination, see Rehydrate data.
- To check the size of your stores, see Store metrics. To remove a store or its data, see Delete or Wipe Store.

9.1 Overview

The storage options of Axoflow (AxoStore on AxoRouter hosts, and AxoLake), and how stores separate your data.

Axoflow provides a range of storage solutions for your security data, with federated search capabilities.

- AxoStore is a lightweight, queryable, temporary storage option that stores data locally on the AxoRouter host. AxoConsole provides federated search for every AxoRouter that has AxoStore deployed, so you can simultaneously run queries on every AxoStore.

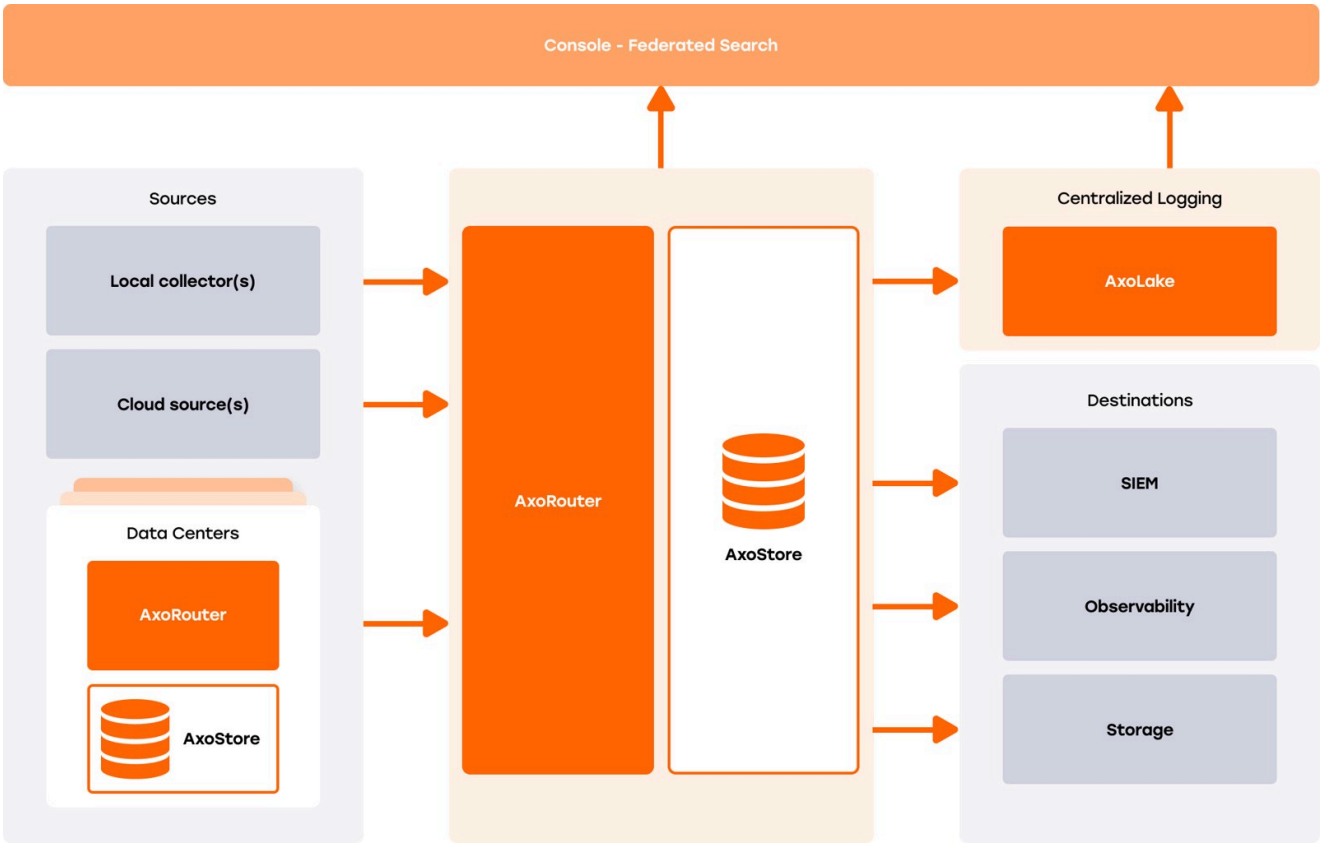
  AxoStore is automatically available for all Axoflow customers, for each AxoRouter node, with 7 days retention, and up to 1TB of storage. For longer retention time or bigger storage, you need an extended AxoStore subscription. Contact us for the details.

  You can retrieve selected data from AxoStore and (rehydrate) it to other destinations for further analysis as needed.

- AxoLake is a tiered SaaS security lake with a scalable hot tier for fast queries, and cost-efficient, S3-compatible cloud-based cold tier. You can use high-level, policy-based routing to decide which data goes to cold storage, hot storage, or your SIEM. If you need to work with data from cold storage, you can rehydrate it into hot storage, or even replay it to send it into your SIEM. Designed for long-term retention, open access, and integration with SIEMs and security analytics.

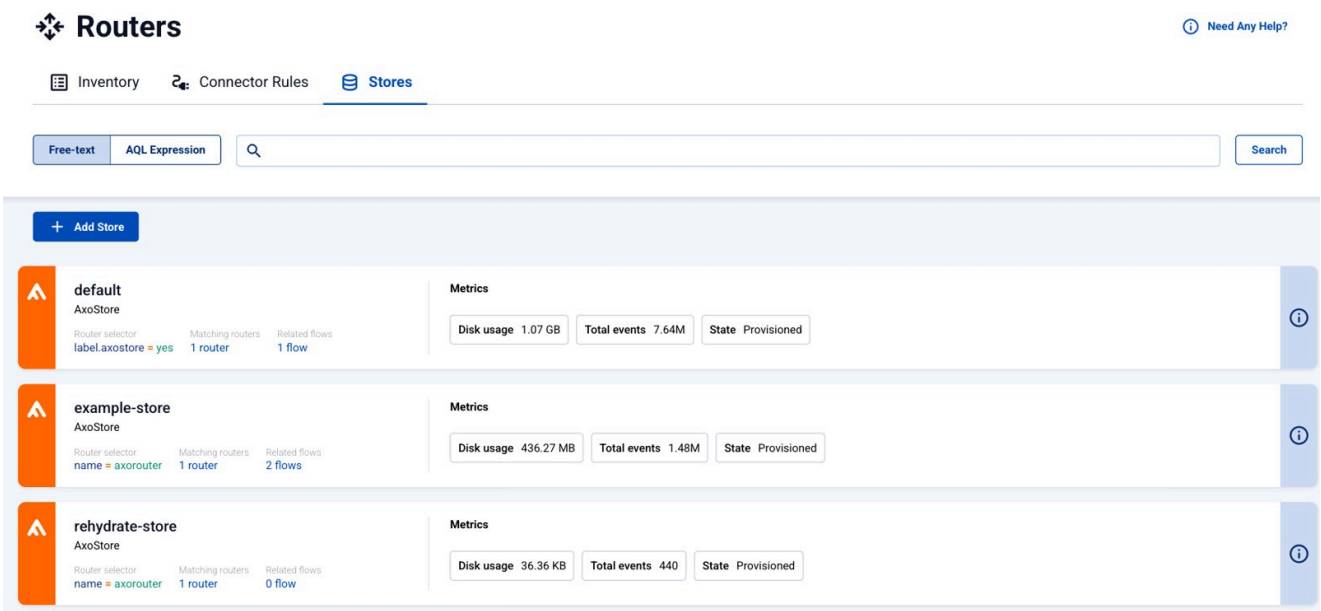
  AxoLake provides cheap, long-term archive storage, while making hot data available for detection and analytics.

To separate the different data in the storage, you can create separate containers called *stores*. Each store appears as separate destination that you can select in your flows.



We refer to stores created in AxoStore as stores, while those in AxoLake are called external stores.

The available stores are listed on the **Routers > Stores** page. To show only one type of stores, select **AQL Expression** and enter `type=axostore` or `type=external`.



Rehydrate data

You can retrieve selected data from your stores and send (rehydrate) it to other destination for further analysis as needed. For details, see Rehydrate data.

9.2 Search

Query AxoStore and AxoLake data with basic or AQL search, check event fields, export the results, or rehydrate them.

The **Search Logs** page provides a federated search console to query all data stored in AxoStores (the storage component of AxoRouter) and external stores (AxoLake). Note that AxoStore is only available on an AxoRouter if it was enabled when the AxoRouter host was deployed or updated. For details, see Install AxoRouter on Linux.

To display data from the specified period without any specific search keywords, just select **Search**.

- **Free-text** mode searches in the values of the following fields of the messages: `body`, `host_name`

  The search is case insensitive. Adding multiple keywords searches for matches in any of the previous fields. This is equivalent to the `@ANY =* keyword1 AND @ANY =* keyword2` AQL query.

- **AQL Expression** mode allows you to search in specific labels of the hosts using different operators.

  It also makes more complex filtering possible, using the Equals, Contains (partial match), and Match (regular expression match) operators. Note that:

  - To execute the search, click **Search**, or hit ESC then ENTER.
  - AxoConsole autocompletes the built-in and custom labels and field names, as well as their most frequent values, but doesn't autocomplete labels and variables created by data parsing and processing steps.
  - You can use the AND and OR operators to combine expressions, and also parenthesis if needed. For details on AQL, see AQL operator reference.
  - The precedence of the operators is the following: parentheses, AND, OR, comparison operators.
  - Use the usual keyboard shortcuts to undo (⌘/Ctrl + Z) or redo (⌘/Ctrl + Shift + Z) your edits.

  For example, `meta.vendor = palo-alto-networks`, `meta.host.labels.location != us-east-1`, or `meta.connection.src_ip =* 192.168`.

  When searching in stores, you can use the following fields (depending on the configuration of the store).

To limit the search, you can:

- Adjust the **Time range** of the search (default is 1 day). Note that AxoStore retains data only for 7 days.
- Search only the data stored on a specific **Router** (by default, AxoConsole searches in the data of every AxoRouter that has AxoStore deployed).
- Search only the data in a specific **Store**. (By default, every AxoRouter that has AxoStore deployed has a `default` store. To create a new store, see Create Store or Create External Store.)

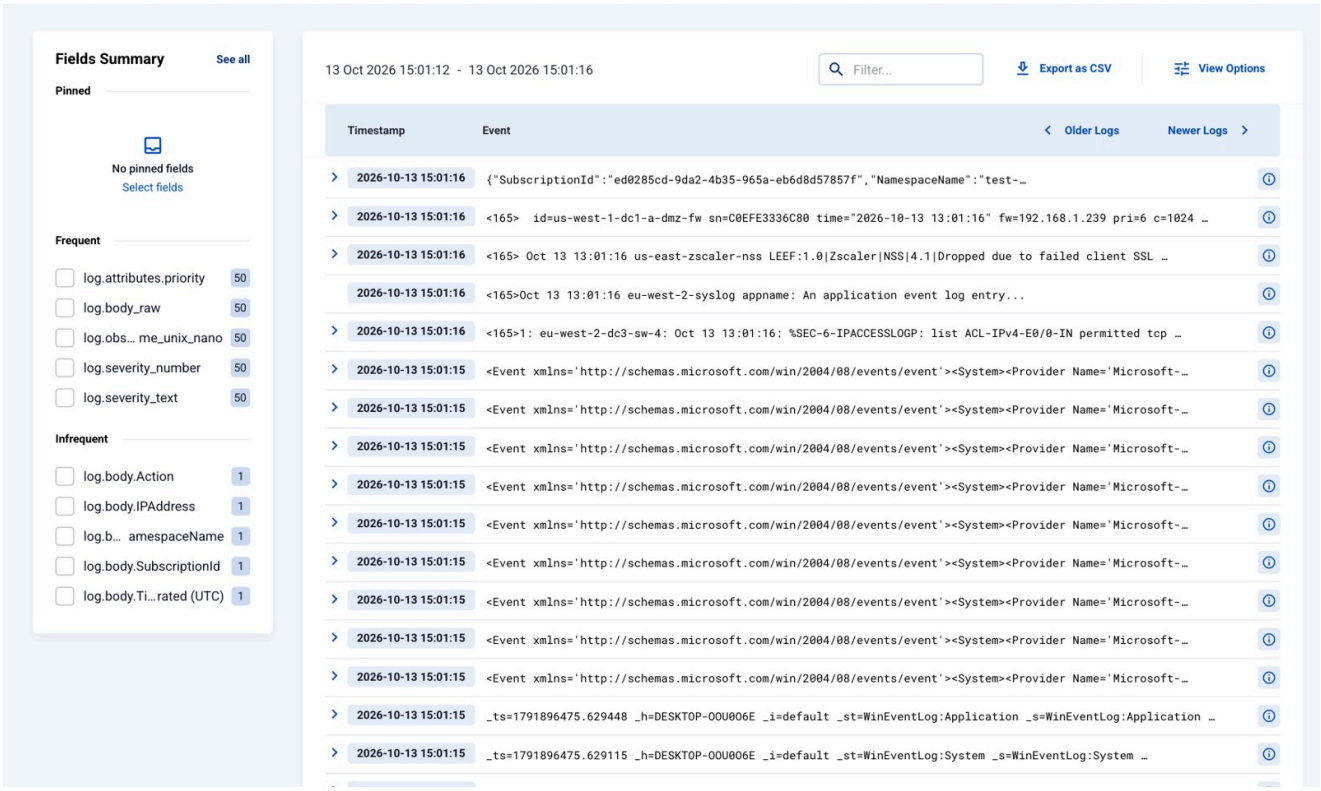


The time distribution of the search results allows you to quickly zoom in on a specific time period. You can select multiple bars to zoom in on them.

Events

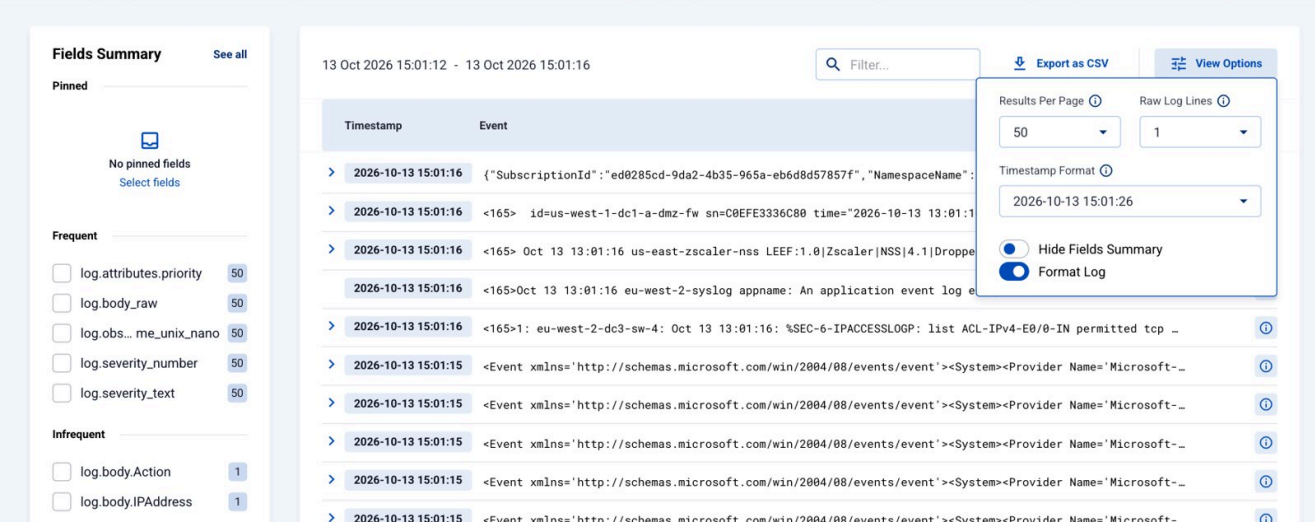
The list below the time distribution shows the search results.

> ## Note
>
> The **search
>
> Filter** field allows you to filter the current page of the search results.



Select **View Options** to customize how the search results are displayed:

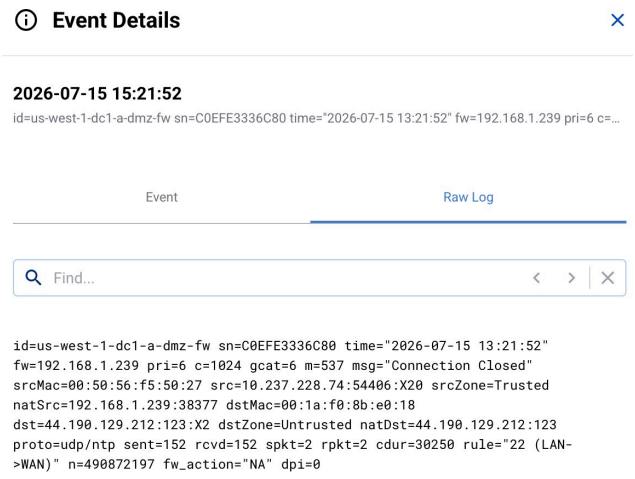
- Adjust the number of **Results Per Page**.
- Increase **Raw Log Lines** to show (wrap) multiple lines of long events
- Switch the **Timestamp Format** between `HH:MM:SS`, `YYYY-MM-DD HH:MM:SS`, and `YYYY-MM-DD HH:MM:SS +-TIMEZONE_OFFSET` (with fractions of a second, for example, milliseconds).
- Enable **Format Log** to format JSON messages into human-readable format.
- Select **Hide Fields Summary** to hide the **Fields Summary** sidebar.



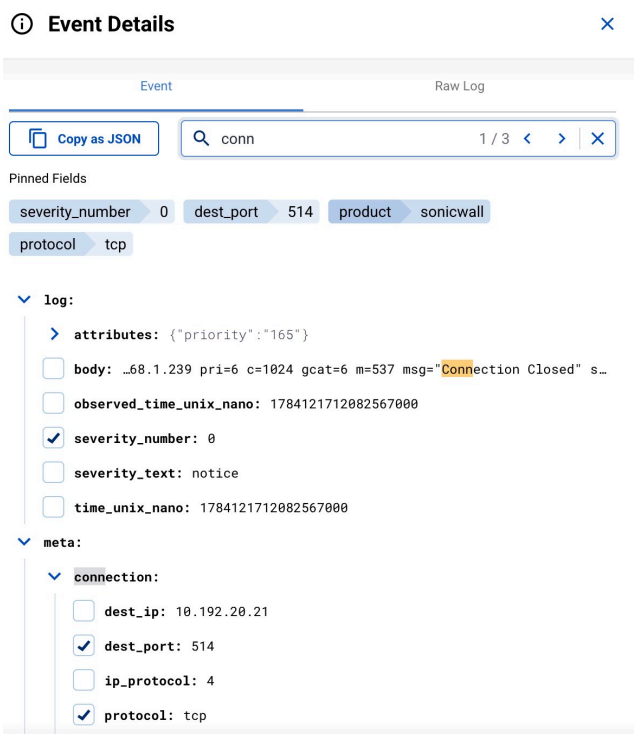
Select the info icon next to an event to see its details.

The **Event Details** panel allows you to inspect a specific event. For details on the available fields and their structure, see Storage schema.

- To display the original log message, select **Raw Log**.

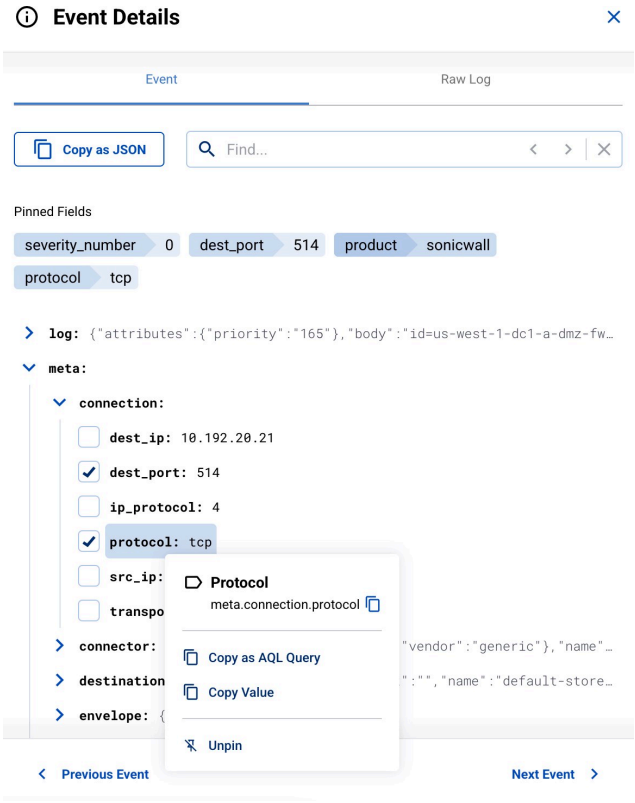


- To copy the entire event as JSON to the clipboard, select **Copy as JSON**.
- To search the event body and metadata, enter a search keyword into the **Find** field. This searches for the string on the **Event Details** panel, including the log, metadata, or label names and values.



- To add a field to the **Pinned Fields** and display it for every event, click the checkbox before the field. Pinned fields show up in the event details, and also in the list of the events, making it easy to get a quick glance on the distribution of the field values. They're included as separate fields when you **Export** the search results into a CSV file.

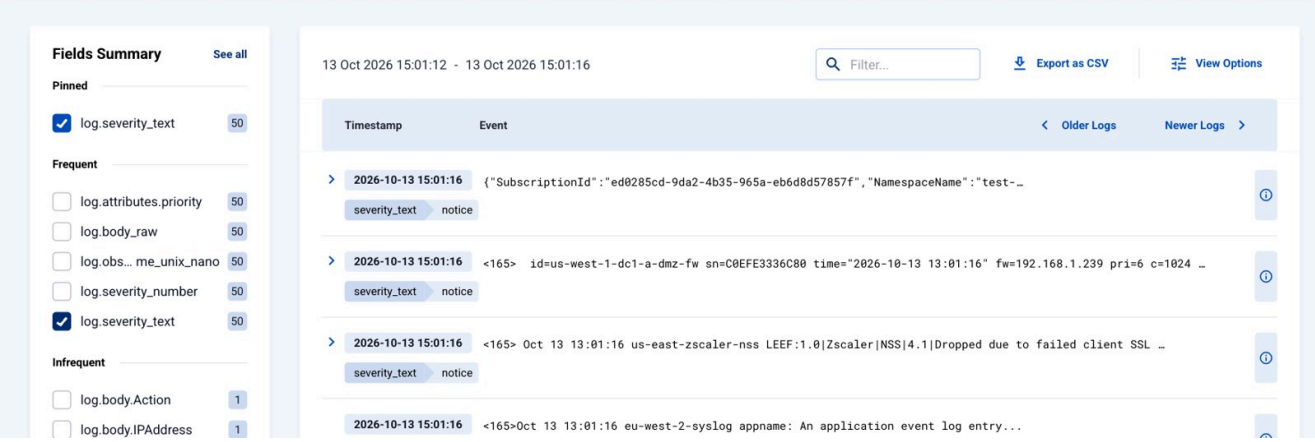
  To unpin a field, select the field in any of the events, then select **Unpin** from the local menu.



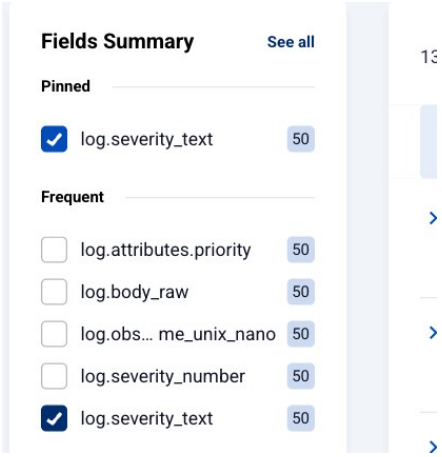
- To copy the name or the value of a field, select the name of the field, then select content_copy or **Copy Value**.
- To copy an AQL query that matches this field and its current value, select the name of the field, then select **Copy as AQL Query**. You can paste this query into the **AQL** search field, and use it for filtering.
- If the field refers to a pipeline element in Axoflow (for example, a flow in the `meta.flow` field, a store, or an AxoRouter), you can jump to the page of the related element.

Fields

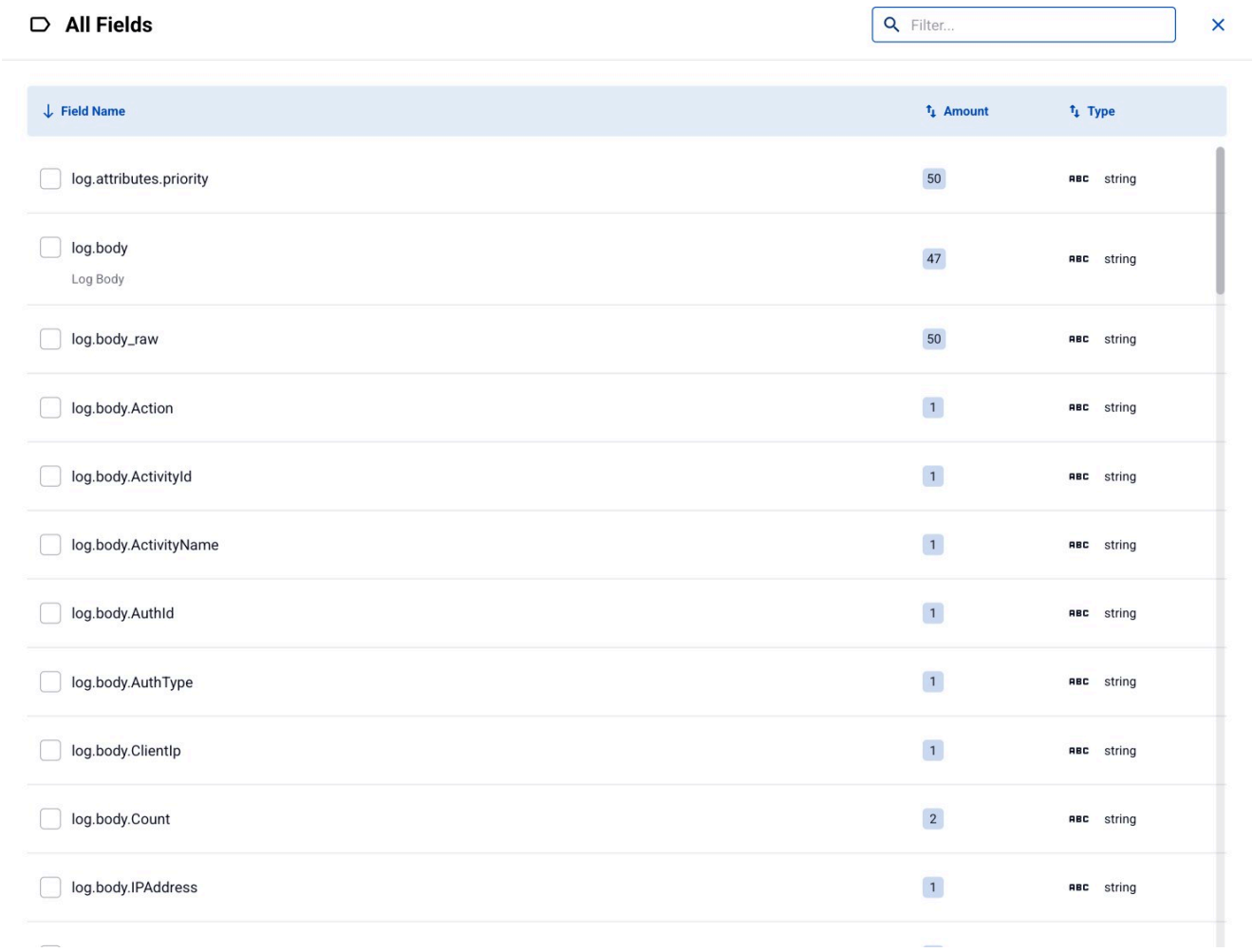
The **Fields Summary** sidebar shows the **Frequent** and **Infrequent** fields that appear in the events of the current page (and the number of their occurrence).



To show a field and its value under each event that has this field, select the checkbox next to the field. The field becomes *pinned*: it appears in the **Pinned** section of the sidebar. To unpin a field, clear its checkbox.



To show other fields (and their occurrence), select **See all** (or **Select fields** if no field is pinned yet), then select the fields you want to pin. Depending on the configuration of the store, you can access the following fields.



Export search results

To export the search results, select **Export as CSV**. Note that:

- Only the search results shown on the current page will be exported (50 entries by default, change **View Options > Results Per Page** to maximum 200). If you need to export even more data, rehydrate the results into a suitable destination.
- By default, the timestamp and the message fields are exported. To include other fields, pin them before exporting the data.

To send the selected data to another destination for further analysis, see Rehydrate data.

Rehydrate search results

You can retrieve the selected data from your stores and send (rehydrate) it to other destination for further analysis as needed. For details, see Rehydrate data.

9.3 Create Store

Create an AxoStore store on the AxoRouter hosts that have AxoStore enabled, and set its retention time.

To create a new store in an AxoStore, complete the following steps.

- AxoStore stores logs locally on the AxoRouter that processed the messages.
- External stores use a separately deployed AxoLake or an existing compatible ClickHouse cluster.
- Logstore is a new store for local and external data storage based on Apache Parquet and Apache Avro. Logstore is now in early-access.

Prerequisites

The AxoRouters you want to store data on must have been provisioned with the **Enable AxoStore** option enabled. (If there is a service called `axostore` in the **Router > your-AxoRouter > Overview > Services** list (even if it's grayed out), that AxoRouter has AxoStore enabled.)

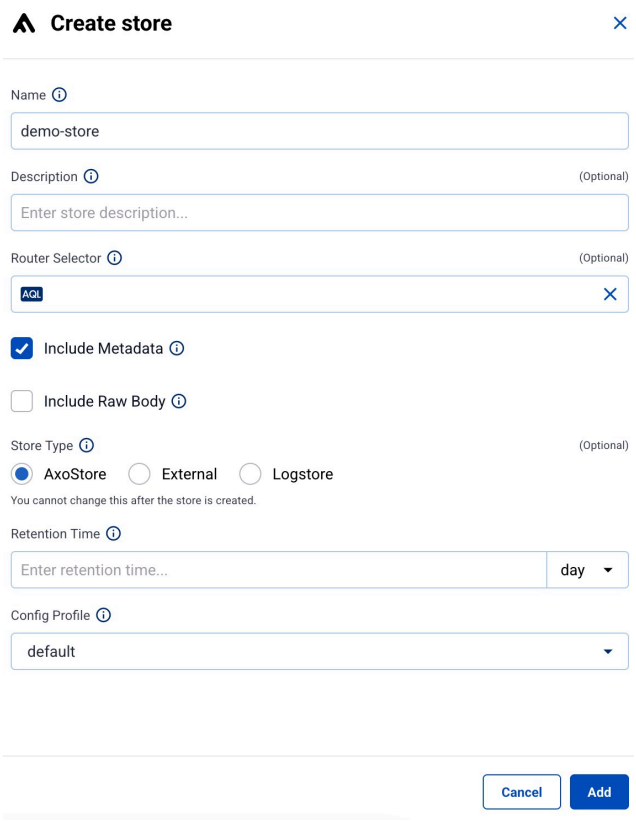
To enable using AxoStore on an already deployed AxoRouter, re-deploy the AxoRouter with the **Enable AxoStore** option enabled.

AxoRouter must have sufficient hardware resources to handle AxoStore. For details and recommendations, see Resource requirements for AxoStore.

Steps

To create a new store in an AxoStore, complete the following steps.

1. Navigate to **Routers > Stores > Add Store**.
2. Enter a **Name** for the store. This name will appear in the list of destinations (with the `-store` suffix) when setting the destination of a Flow. You cannot modify the name of the store later.



3. (Optional) Enter a description for the store.

4. Set **Store Type** to **AxoStore**.

5. Set the **Router Selector** so it matches the AxoRouter instances where you want to create this store.

   You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

   - If you leave the router selector empty, it matches every AxoRouter instance.
   - To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.
   - If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

6. (Optional) Select additional options about which message parts to store:

   - **Include Metadata**: If enabled, the metadata of the events is stored as a separate object. Otherwise, only the metadata that has dedicated fields in the Storage schema will be recorded.

   - **Include Raw Body**: Store also the original, unparsed (raw) message as a separate field.

     CAUTION:

> Enabling this option approximately doubles the storage size of the message.

7. Set the **Retention Time** (in days or months) so older data is automatically deleted from the store. You cannot modify this parameter later.

8. Select the **Config Profile** to use for the AxoStore. The profile determines resource limits (CPU, memory) and various optimization and performance settings. Every AxoRouter that creates this AxoStore will use the same profile. The following profiles are available:

   > **CAUTION:**
   >
   > You cannot change the **Config Profile** of the AxoStore later.

   - **Default**: Recommended for small-scale deployments and evaluation use cases that receive about 1000 events per second.
   - **Large**: For AxoRouters with powerful hardware (at least 16 vCPU and 32GiB RAM), used for storing large amount of data.

   For AxoRouter hardware sizing recommendations, see Resource requirements.

9. Select **Add**.

10. The new store will be created on the AxoRouters matching the **Router Selector**. You can check the stores available on an AxoRouter by selecting **Routers > your-AxoRouter > Overview > Stores**.

11. (Optional) By default, only users with the **log-searcher** role can search in the stores. To grant access to a specific store, see Store access roles.

9.4 Create External Store

Create a store in AxoLake or a self-managed ClickHouse instance, and use it as a destination in your flows.

To create a new store in an AxoLake or a self-managed ClickHouse instance, complete the following steps. (For details on deploying AxoLake or an appropriate self-managed ClickHouse instance, contact our support team.)

- AxoStore stores logs locally on the AxoRouter that processed the messages.
- External stores use a separately deployed AxoLake or an existing compatible ClickHouse cluster.
- Logstore is a new store for local and external data storage based on Apache Parquet and Apache Avro. Logstore is now in early-access.

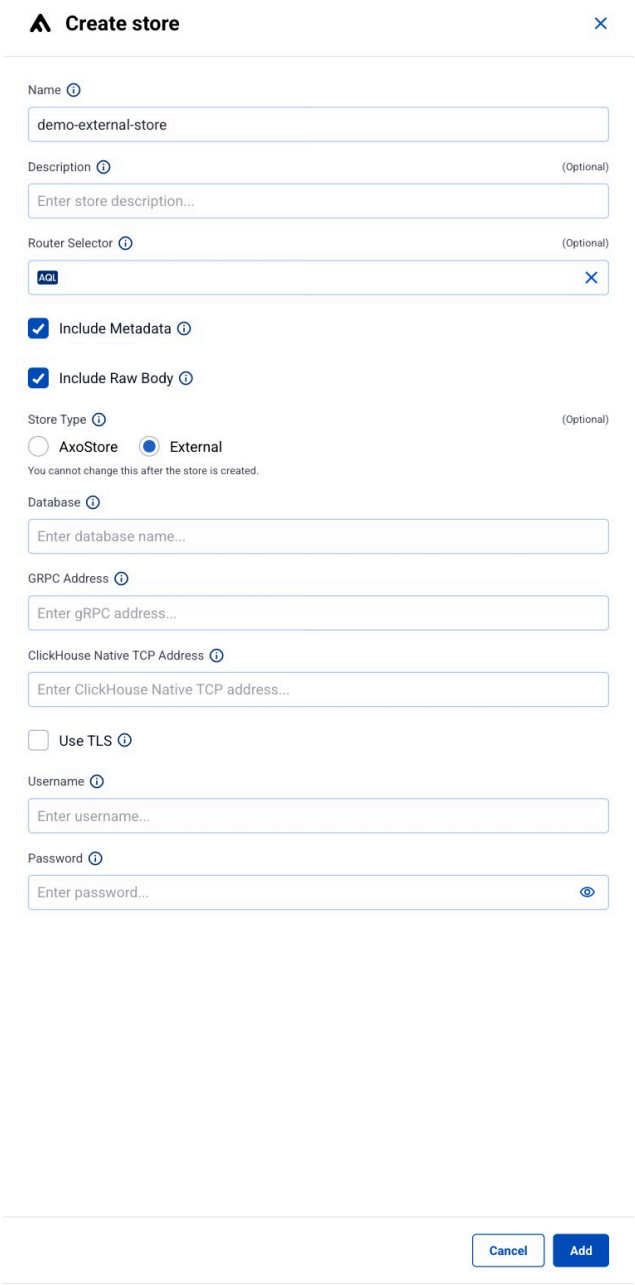
Prerequisites

- You must have an AxoLake or a self-managed ClickHouse instance available.
- You'll need user credentials (username and password) to authenticate to the database. The account must have:
  - `INSERT` privileges for the database where you're creating the store to send data, and
  - `SELECT` privileges to access the data from the **Search Logs** page.
- Your AxoRouter instances must be able to access the database on the network level.

Steps

To create a new store in an AxoLake, complete the following steps.

1. Navigate to **Routers > Stores > Add Store**.
2. Enter a **Name** for the store. This name will appear in the list of destinations (with the `-store` suffix) when setting the destination of a Flow. You cannot modify the name of the store later.



3. (Optional) Enter a description for the store.

4. (Optional) Select additional options about which message parts to store:

   - **Include Metadata**: If enabled, the metadata of the events is stored as a separate object. Otherwise, only the metadata that has dedicated fields in the Storage schema will be recorded.

   - **Include Raw Body**: Store also the original, unparsed (raw) message as a separate field.

     > **CAUTION:**
     >
     > Enabling this option approximately doubles the storage size of the message.

5. Set **Store Type** to **External**.

6. Enter the name of the **Database** to create the store in. (This is the name of the database you've created in AxoLake/ClickHouse. It can only contain letters, numbers, and underscores.)

7. Set the **GRPC Address** and the **ClickHouse Native TCP Address** of the database listener, for example, `10.0.0.123:9100` and `10.0.0.123:9000`. Instead of IP address, you can also use FQDN.

8. Select **Use TLS** to encrypt the connection to the external store.

9. Enter the credentials of the user account used to access the database into the **Username** and **Password** fields.

10. Select **Add**.

11. (Optional) By default, only users with the **log-searcher** role can search in the stores. To grant access to a specific store, see Store access roles.

9.5 Create Logstore

Create a Logstore (early access), which keeps data in Parquet on AxoRouter hosts and in external object storage.

> **Note**
>
> **Logstore is currently an early-access feature and not enabled on every tenant. If you want to test it, contact our support team.**

- AxoStore stores logs locally on the AxoRouter that processed the messages.
- External stores use a separately deployed AxoLake or an existing compatible ClickHouse cluster.
- Logstore is a new store for local and external data storage based on Apache Parquet and Apache Avro. Logstore is now in early-access.

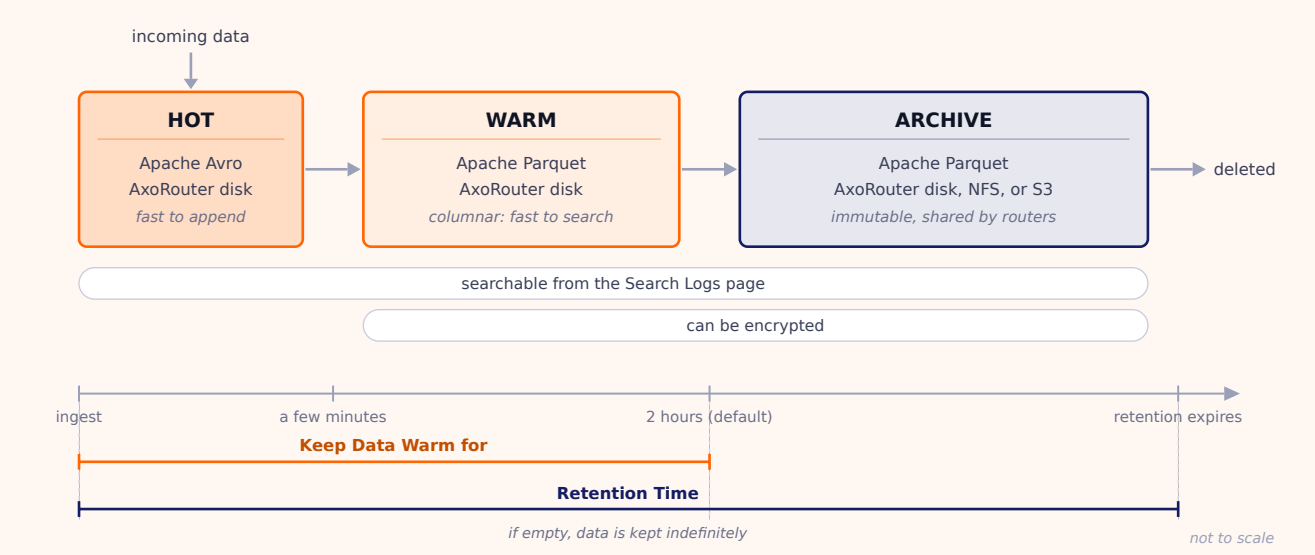
Logstore tiers

A Logstore keeps your data in three tiers. Data enters the hot tier, then moves through the warm tier into the archive. You can search every tier from the **Search Logs** page, no matter which tier currently holds the data you're looking for.

- **Hot tier**: Receives the data as it arrives, stores it in Apache Avro format, on the local disk of the AxoRouter. Avro is a row-oriented format that's cheap to append to, which is what a tier taking live data needs. Data spends only a few minutes here before it moves to the warm tier.
- **Warm tier**: Holds recent data in Apache Parquet format, also on the local disk of the AxoRouter. Parquet is column-oriented, so a search reads only the fields it needs. Data stays warm for 2 hours by default, then moves to the archive.
- **Archive tier**: Holds the data for the long term, also in Parquet format. Depending on the type of the archive, the files are stored on the local disk of the AxoRouter, on an NFS mount, or on Amazon S3 (or another S3-compatible service). Archive files are immutable, and several AxoRouters can write to the same remote archive. The Logstore deletes archived data when its retention time expires, or keeps it indefinitely if you don't set one.

You can encrypt the warm and archive tiers. The hot tier isn't encrypted by default, because data passes through it in a few minutes.

When querying data from a Logstore, the `log.attributes.log.source` label shows which tier the data is in.



Prerequisites

- AxoRouter prerequisites:
  - The AxoRouters you want to store data on must have been provisioned with the **Enable Logstore** option enabled. (If there is a service called `axorouter-logstore` in the **Router > your-AxoRouter > Overview > Services** list (even if it's grayed out), that AxoRouter has Logstore enabled.)

    To enable using Logstore on an already deployed AxoRouter, re-deploy the AxoRouter with the **Enable Logstore** option enabled.
  - AxoRouter must have sufficient hardware resources to handle Logstore, especially disk space for the hot and warm tiers. To place the Logstore tiers on separate disks, set the Logstore directories when you deploy the AxoRouter. For sizing recommendations, contact our support team.
- If you want to encrypt the store, you'll need a 256-bit AES key (as 64 hexadecimal characters). For example, you can generate it with the OpenSSL command-line tool: `openssl rand -hex 32`
- For using S3 archiving:

  An existing S3 bucket configured for programmatic access, and the related `ACCESS_KEY` and `SECRET_KEY` of a user that can access it. Logstore writes, reads, and deletes the archived objects, so the user needs the following permissions:
  - `s3:ListBucket`
  - `s3:GetObject`
  - `s3:PutObject`

- `s3:DeleteObject`
- `s3:AbortMultipartUpload`

If the default encryption of the bucket is SSE-KMS (`aws:kms`), the AWS role or user also needs the `kms:GenerateDataKey` and `kms:Decrypt` permissions on the KMS key. (For details on why `kms:Decrypt` is mandatory, check this AWS Knowledge Center entry.)

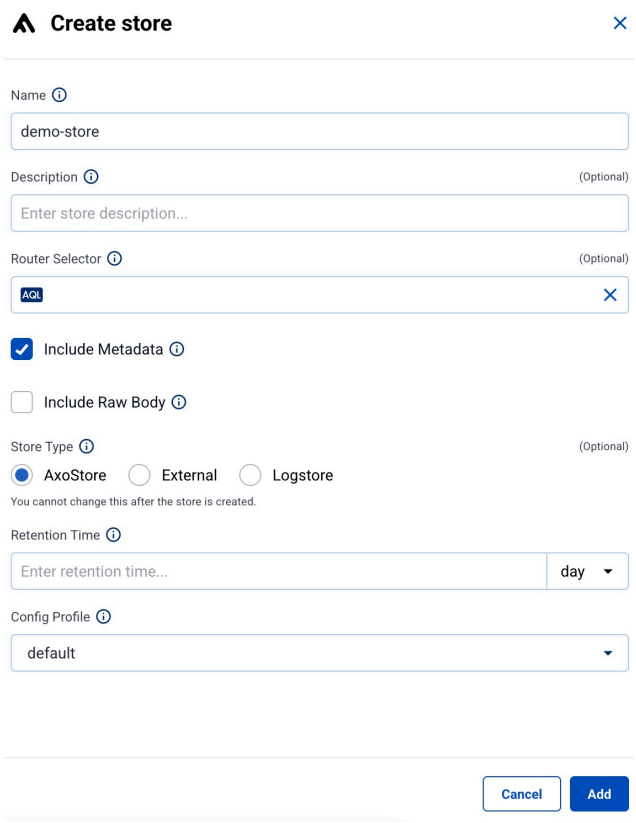
See a sample IAM policy

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Sid": "ListLogstorePrefix",
        "Effect": "Allow",
        "Action": "s3:ListBucket",
        "Resource": "arn:aws:s3:::<BUCKET>",
        "Condition": {
          "StringLike": { "s3:prefix": "<PREFIX>/*" }
        }
      },
      {
        "Sid": "ReadWriteLogstoreObjects",
        "Effect": "Allow",
        "Action": [
          "s3:GetObject",
          "s3:PutObject",
          "s3:DeleteObject",
          "s3:AbortMultipartUpload"
        ],
        "Resource": "arn:aws:s3:::<BUCKET>/<PREFIX>/*"
      },
      {
        "Sid": "UseBucketKmsKey",
        "Effect": "Allow",
        "Action": ["kms:GenerateDataKey", "kms:Decrypt"],
        "Resource": "arn:aws:kms:<REGION>:<ACCOUNT_ID>:key/<KEY_ID>"
      }
    ]
  }
```

Create Logstore

To create a new Logstore, complete the following steps.

1. Navigate to **Routers > Stores > Add Store**.
2. Enter a **Name** for the store. This name will appear in the list of destinations (with the `-store` suffix) when setting the destination of a Flow. You cannot modify the name of the store later.



3. (Optional) Enter a description for the store.
4. Set **Store Type** to **Logstore**.
5. Set the **Router Selector** so it matches the AxoRouter instances where you want to create this store.

   You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

   - If you leave the router selector empty, it matches every AxoRouter instance.
   - To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.
   - If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

6. (Optional) Select additional options about which message parts to store:

   - **Include Metadata**: If enabled, the metadata of the events is stored as a separate object. Otherwise, only the metadata that has dedicated fields in the Storage schema will be recorded.

   - **Include Raw Body**: Store also the original, unparsed (raw) message as a separate field.

     > **CAUTION:**
     >
     > Enabling this option approximately doubles the storage size of the message.

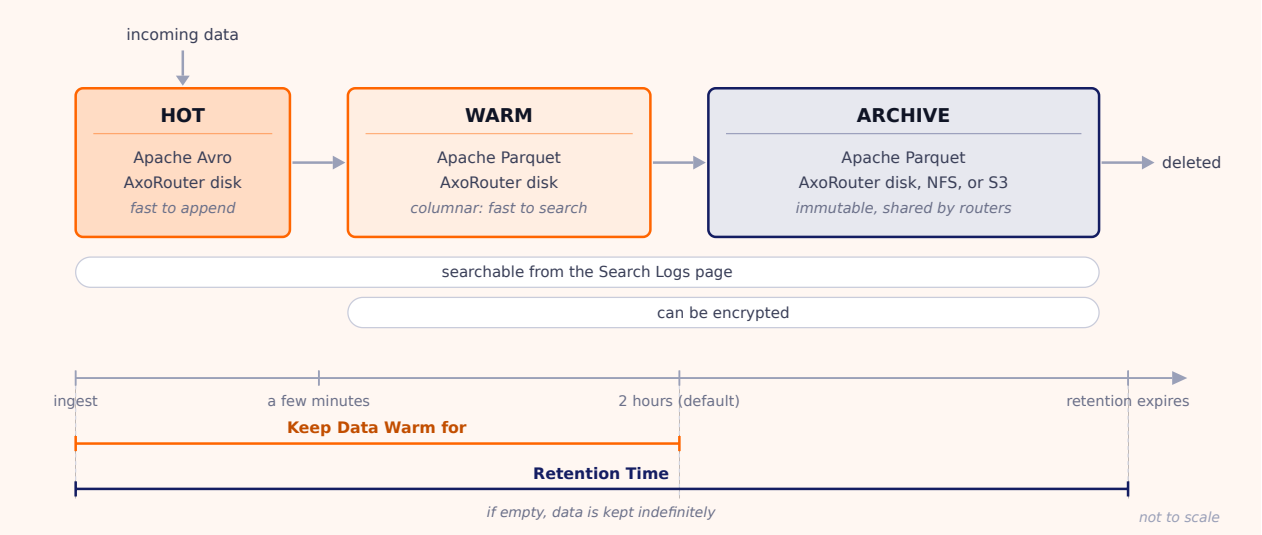
7. Set how much time data spends in the different storage tiers.

   - **Retention Time** (in hours or days) configures how long the data is kept in the Logstore, measured from the timestamp when the data was generated. After this period the data is automatically deleted from the store. If **Retention Time** is empty, the data is kept indefinitely.

     > **Note**
     >
     > **The timestamp when the data is generated and when AxoRouter receives it are usually close to each other. In some cases the difference can be important, for example, if the retention time is low and the incoming data is older (for example, data is received after a long network outage), or when you're rehydrating old data into a logstore that has low retention time.**

   - **Keep Data Warm for** (in hours or days) configures how long the data is kept in the warm tier, measured from the timestamp when the data was generated. After this period the data is moved to the archive. Default value: 2 hours. The value of **Keep Data Warm for** can't be longer than the **Retention Time**.

     > **Note**
     >
     > **The timestamp when the data is generated and when AxoRouter receives it are usually close to each other. In some cases the difference can be important, for example, if the keep data warm value is low and the incoming data is older (for example, data is received after a long network outage), or when you're rehydrating old data into a logstore that has keep data warm value.**



8. (Optional) To encrypt the store, enter a 256-bit AES key (as 64 hexadecimal characters) into the **Encryption Key** field. Keep a copy of the key, without it the data cannot be read.

   > **CAUTION:**
   >
   > You cannot change encryption settings later. Encryption applies to the warm and archive tiers of the data storage. The hot tier (which is temporal and the data usually spends only about five minutes) isn't encrypted by default.

9. Select the **Type** of the archive.

   > **CAUTION:**
   >
   > You cannot change the type of the archive later.

   - **Local**: Store archived data locally on AxoRouter. The location of the archive files is `/var/lib/logstore/archive/<store-name>` by default (can be set when AxoRouter is deployed). If needed, you can override it using the **Archive Path** field.
   - **NFS**: Store archived data on the NFS mount configured when AxoRouter was deployed. If needed, you can override the location of the archive directory using the **Archive Path** field.

     > ## Note
     >
     > **Archiving data to an NFS mount is supported only for on-premises and air-gapped AxoConsole deployments.**

   - **S3**: Archive the data to Amazon S3, or to another S3-compatible service. Configure access to the S3 service:

     1. **Bucket name**: The name of the bucket you want to use.

2. **Prefix**: The prefix path within the S3 bucket.
3. Select the **Region** of the bucket, or enter a custom endpoint URL into the **Endpoint** field.
4. Enter the **Access key** and the **Secret key** for the account you want to use.

10. Select **Add**.

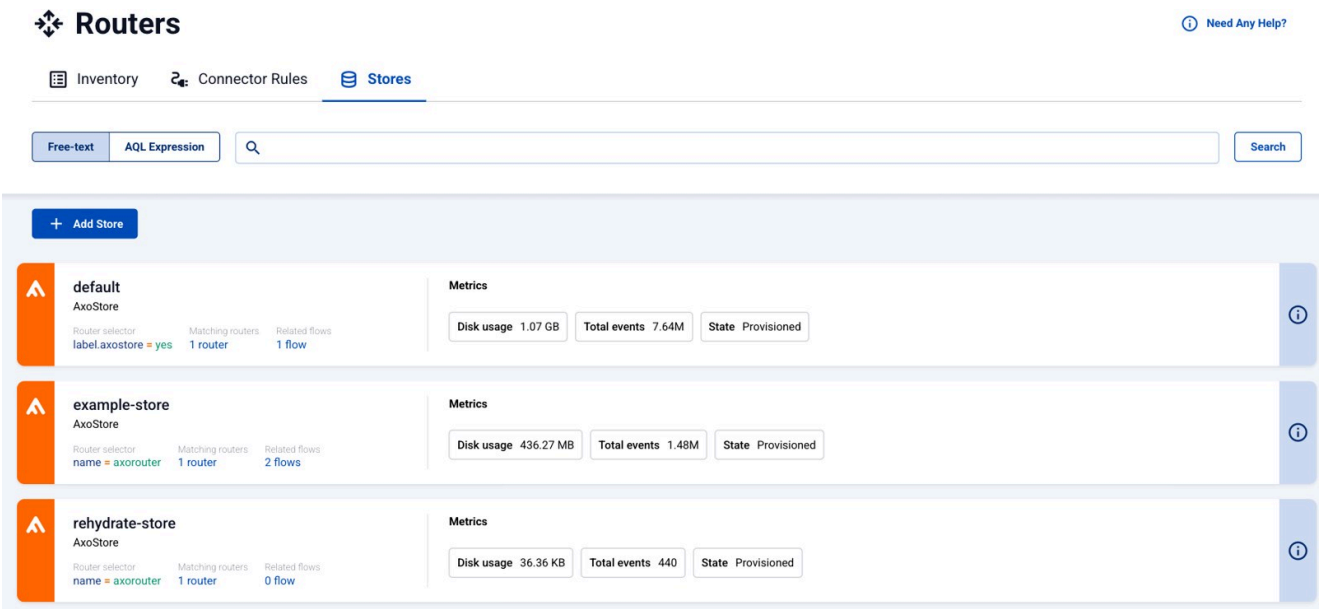
11. The new store will be created on the AxoRouters matching the **Router Selector**. You can check the stores available on an AxoRouter by selecting **Routers > your-AxoRouter > Overview > Stores**.

12. (Optional) By default, only users with the **log-searcher** role can search in the stores. To grant access to a specific store, see Store access roles.

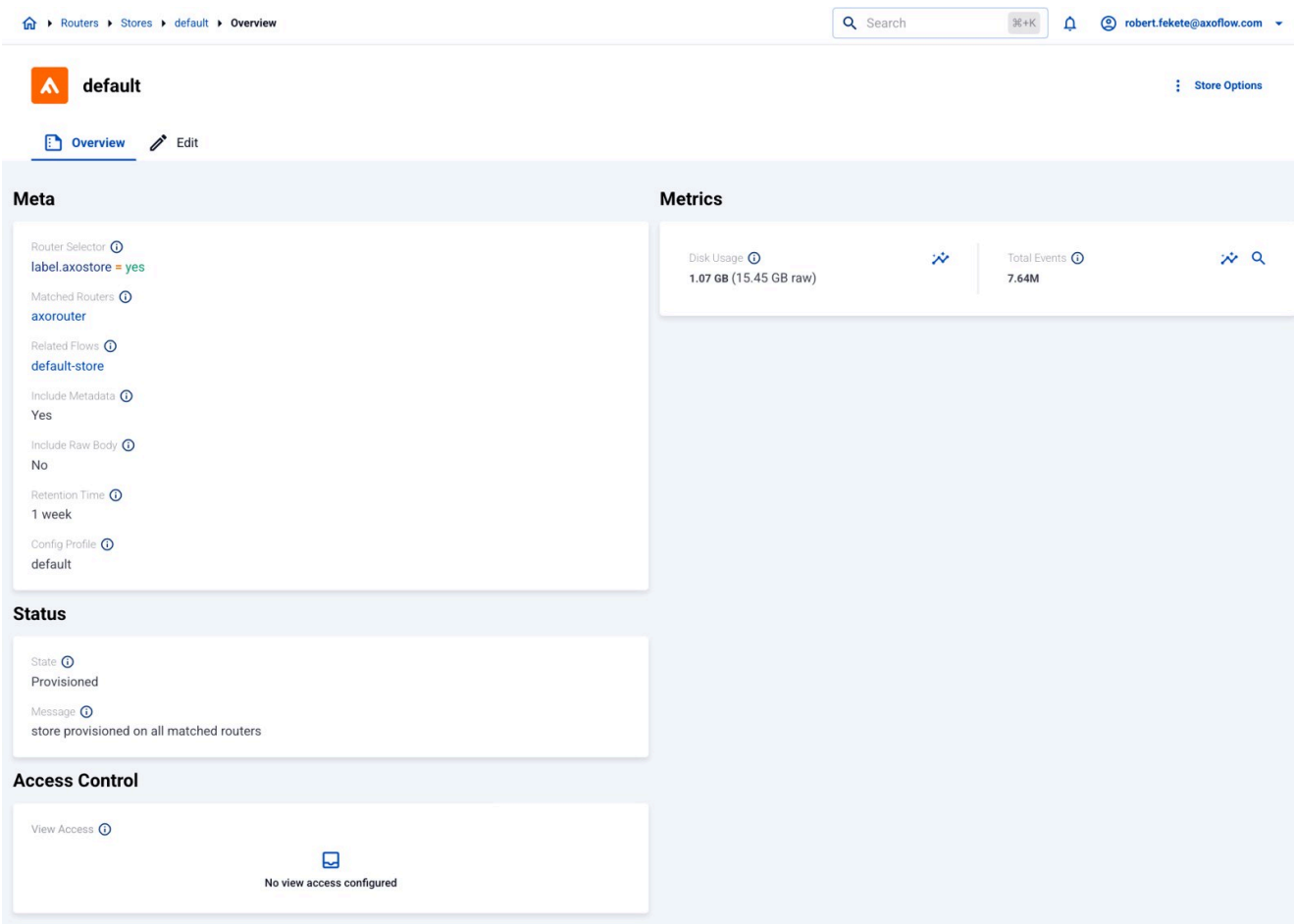
9.6 Store metrics

Check the event count and disk usage of each store, and open its analytics and search.

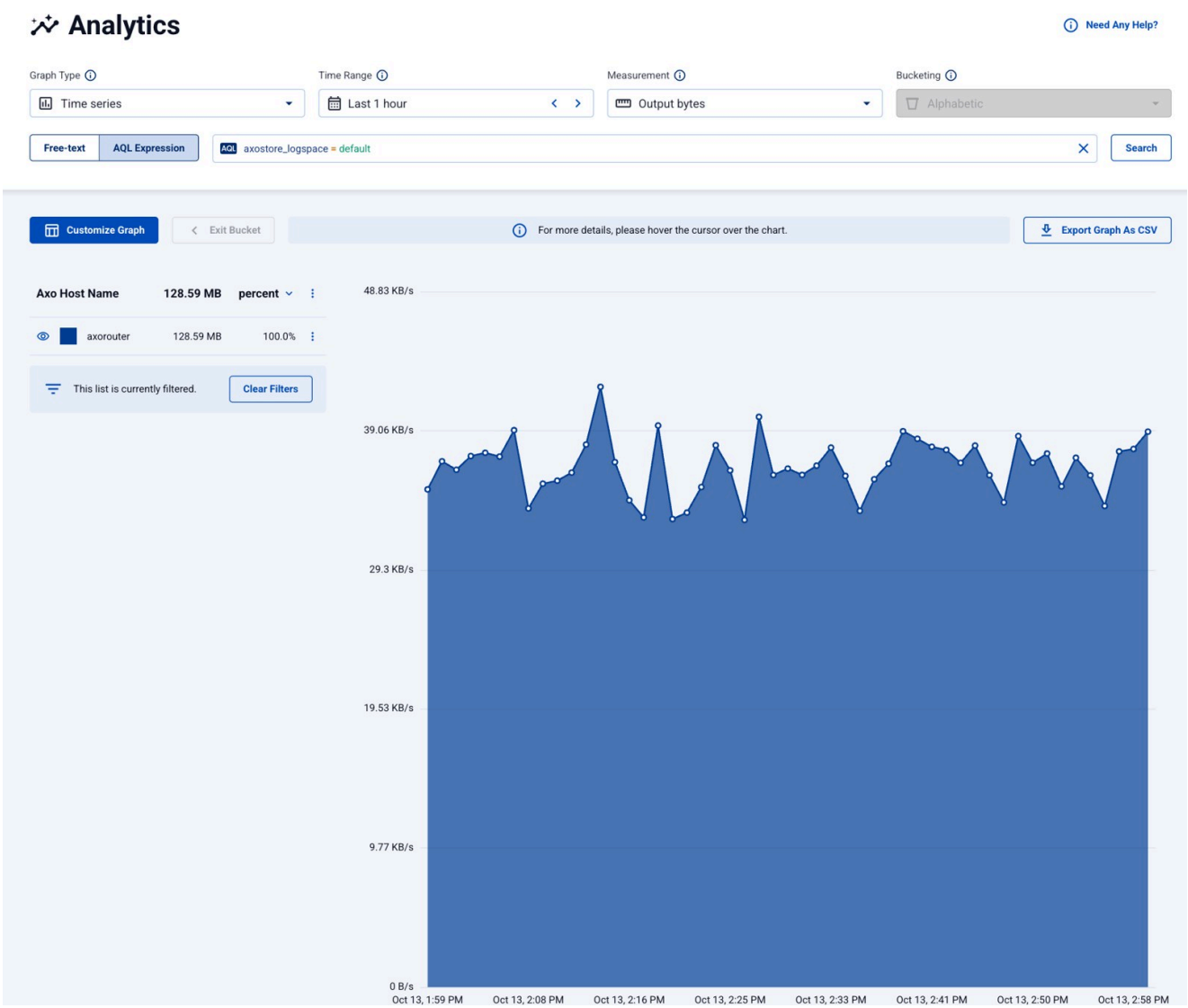
The **Routers > Stores** page lists the available stores. The list also shows the basic metrics of each store, including the number of events and the amount of disk used that the store uses.



For more details, select the name of a specific store.



Select insights to see the analytics (for example, the time series chart) of the store metrics.



To search the stores, select search . Selecting search in the metrics of a particular AxoRouter automatically filters the search to that AxoRouter.

9.7 Delete or Wipe Store

The difference between deleting and wiping a store, and how to do each.

There are two ways to remove a store: delete and wipe.

- Deleting a store immediately terminates the connection between your AxoRouters and the store. No further data will be added to the store. Data already in the store will be automatically deleted after the configured retention time of the store expires. To recover data from a deleted store before the retention time expires, contact our support team.
- Wiping instantly deletes all data in the store.

> **CAUTION:**
>
> Wiping a store is irreversible and cannot be undone. All data in the store is lost forever.
>
> Wiping Logstores is currently not supported.

To remove a store, complete the following steps.

1. Navigate to **Routers > Stores**. Alternatively, select `⌘/Ctrl + K` and enter the name of the store.
2. Select the store you want to remove, then:
   - To delete the store, select ⋮ **Store Options > Delete**.
   - To wipe the store, select ⋮ **Store Options > Wipe data**.
3. Confirm that you want to remove the store.

9.8 Rehydrate data

Send data from your stores to a SIEM or another destination.

You can retrieve selected data from your stores and send (rehydrate) it to other destination for further analysis as needed. For details, see Rehydrate data.

10 Data management

How Axoflow processes your security data, and how to configure, route, and monitor data flows and their processing steps.

In Axoflow, a Flow selects a set of data, processes it, and sends it to a destination. AxoConsole converts your flows into the configuration of your AxoRouter instances, and deploys it.

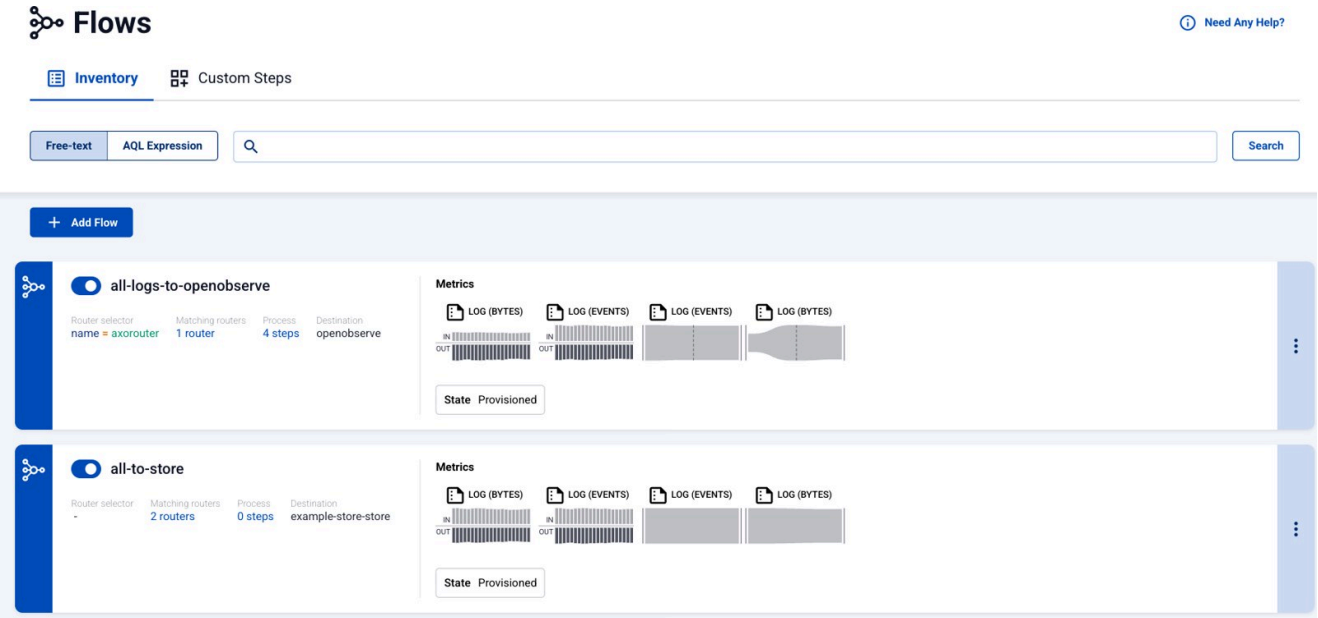
- To understand the parts of a flow, see Flow overview. To create or disable a flow, see Manage flows.
- To filter, parse, reduce, and change the data in a flow, see Processing steps. To create a parser for unknown log messages, see AI assistant.
- To test a flow, see Flow tapping. To check how much data a flow processes, see Flow metrics and Flow analytics.
- To send stored data to a destination again, see Rehydrate data.

For the path data takes through Axoflow, see Processing elements.

10.1 Flow overview

The parts of a flow (router selector, processing steps, destination), how to filter the Flows page, and what the fallback flow does.

Axoflow uses *flows* to manage the routing and processing of security data. A flow applies to one or more AxoRouter instances. The **Flows > Inventory** page lists the configured flows, and also highlights if any alerts apply to a flow. To turn a flow off or on, use the toggle next to its name.

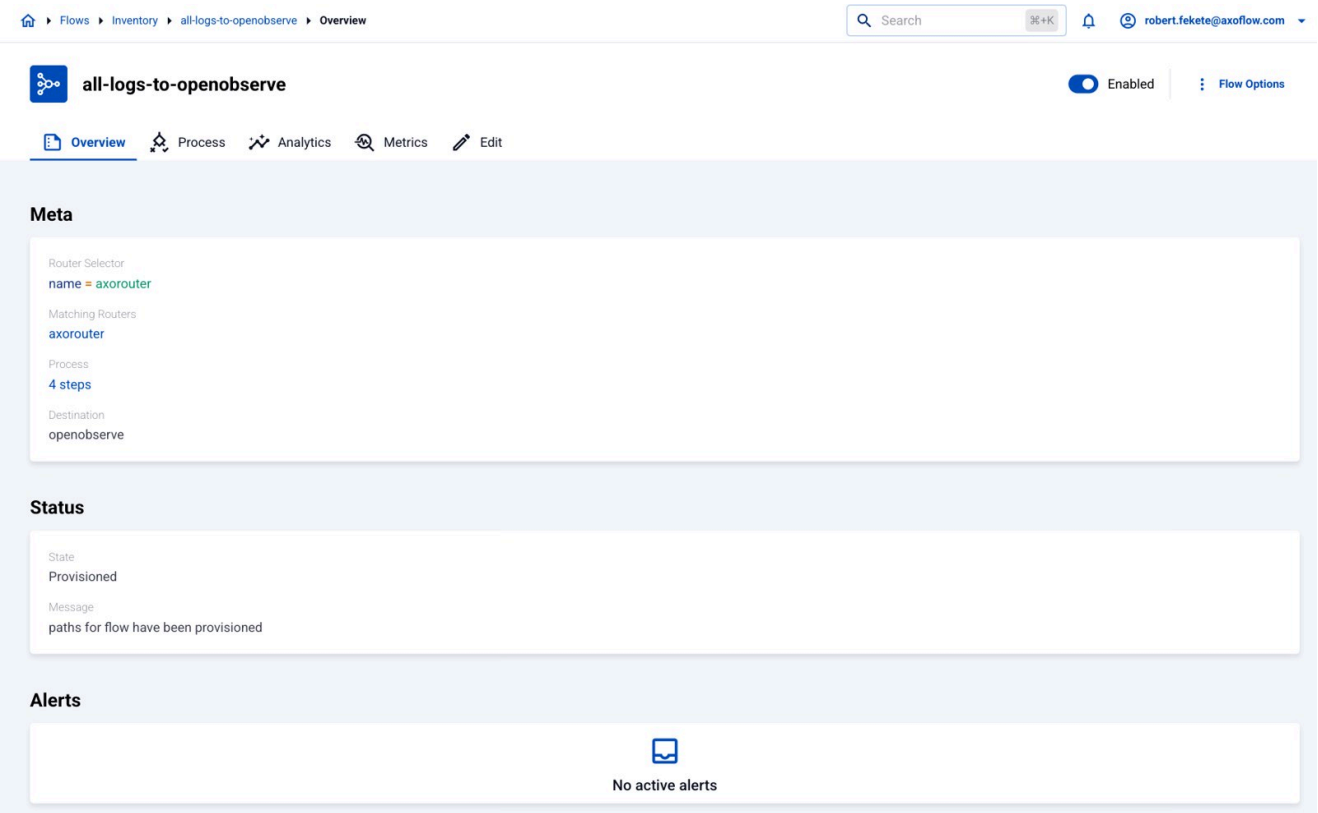


Each flow consists of the following main elements:

- A **Router selector** that specifies the AxoRouter instances the flow applies to. Multiple flows can apply to a single AxoRouter instance.
- **Processing steps** that filter and select the messages to process, set/unset message fields, and perform different data transformation and data reduction.
- A **Destination** where the AxoRouter instances of the flow deliver the data. Destinations can be external destinations (for example, a SIEM), stores or external stores, or other AxoRouter instances.

Based on the flows, AxoConsole automatically generates and deploys the configuration of the AxoRouter instances. To display the details of a flow, select its name, or select more_vert > **Overview** on its card. To create a new flow based on an existing one, select more_vert > **Clone** on its card. AxoConsole opens the **Clone flow** form with the settings of the original flow.

To reuse a sequence of processing steps in multiple flows, create a custom step on the **Flows > Custom Steps** page.



Filter flows

To find or display only specific flows, you can use the filter bar.

- **Free-text** mode searches in the following fields of the flow: **Name**, **Destination**, **Description**.

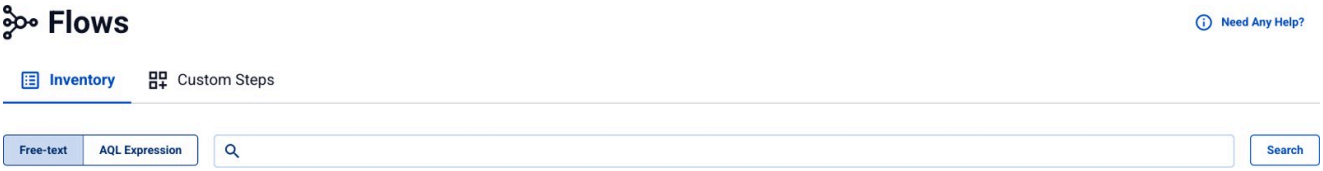
  The search is case insensitive. Adding multiple keywords searches for matches in any of the previous fields. This is equivalent to the `@ANY =* keyword1 AND @ANY =* keyword2` AQL query.

- **AQL Expression** mode allows you to search in specific fields of the flows.

  It also makes more complex filtering possible, using the Equals, Contains (partial match), and Match (regular expression match) operators. Note that:

  - To execute the search, click **Search**, or hit ESC then ENTER.
  - AxoConsole autocompletes the built-in and custom labels and field names, as well as their most frequent values, but doesn't autocomplete labels and variables created by data parsing and processing steps.
  - You can use the AND and OR operators to combine expressions, and also parenthesis if needed. For details on AQL, see AQL operator reference.
  - The precedence of the operators is the following: parentheses, AND, OR, comparison operators.

- Use the usual keyboard shortcuts to undo (⌘/Ctrl + Z) or redo (⌘/Ctrl + Shift + Z) your edits.



Fallback flow

Fallback flows process messages that weren't processed by other, 'non-fallback' flows. If configured, the fallback flow acts as a default flow: for example, you can use it to send all data to a low-cost storage that other flows haven't sent to other destinations.

> **Note**
>
> **You can configure multiple fallback flows, but only one fallback flow can apply to an AxoRouter (so the Router Selector of the fallback flows can't overlap).**



10.2 Manage flows

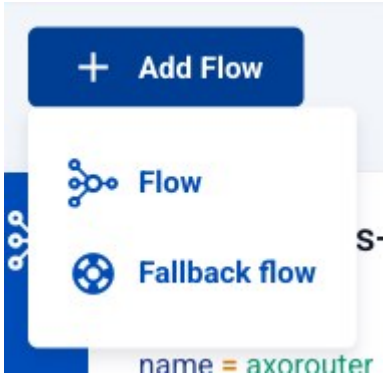
Create a flow that selects, processes, and routes data to a destination, or disable a flow without deleting it.

Create flow

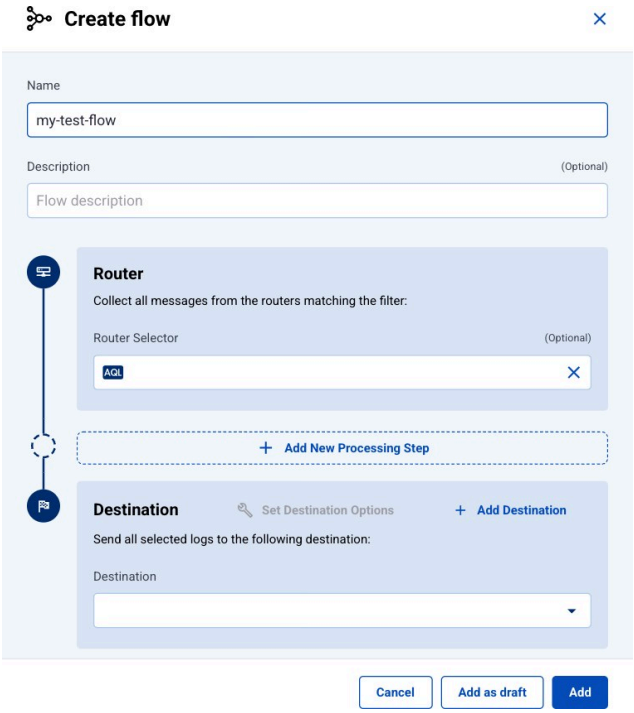
To create a new flow, open the AxoConsole, then complete the following steps:

1. Select **Flows**.
2. Select **Add Flow > Flow**.

   To create a fallback flow, select **Add Flow > Fallback flow**.



3. Enter a name for the flow, for example, `my-test-flow`.



4. In the **Router Selector** field, enter an expression that matches the routers you want to apply the flow to. To select a specific router, use a name selector, for example, `name = my-axorouter-hostname`.

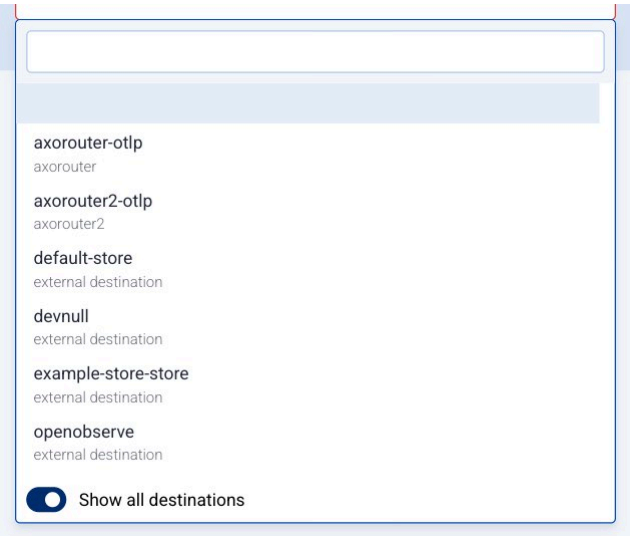
You can use any labels and metadata of the AxoRouter hosts in the Router selectors, for example, the hostname of the AxoRouter, or any custom labels.

- If you leave the router selector empty, it matches every AxoRouter instance.
- To select only a specific AxoRouter instance, set the `name` field to the name of the instance as selector. For example, `name = my-axorouter`.
- If you set multiple fields in the selector, the selector will match only AxoRouter instances that match all elements of the selector. (There in an AND relationship between the fields.)

> **Note**
>
> **You can configure multiple fallback flows, but only one fallback flow can apply to an AxoRouter (so the Router Selector of the fallback flows can't overlap).**

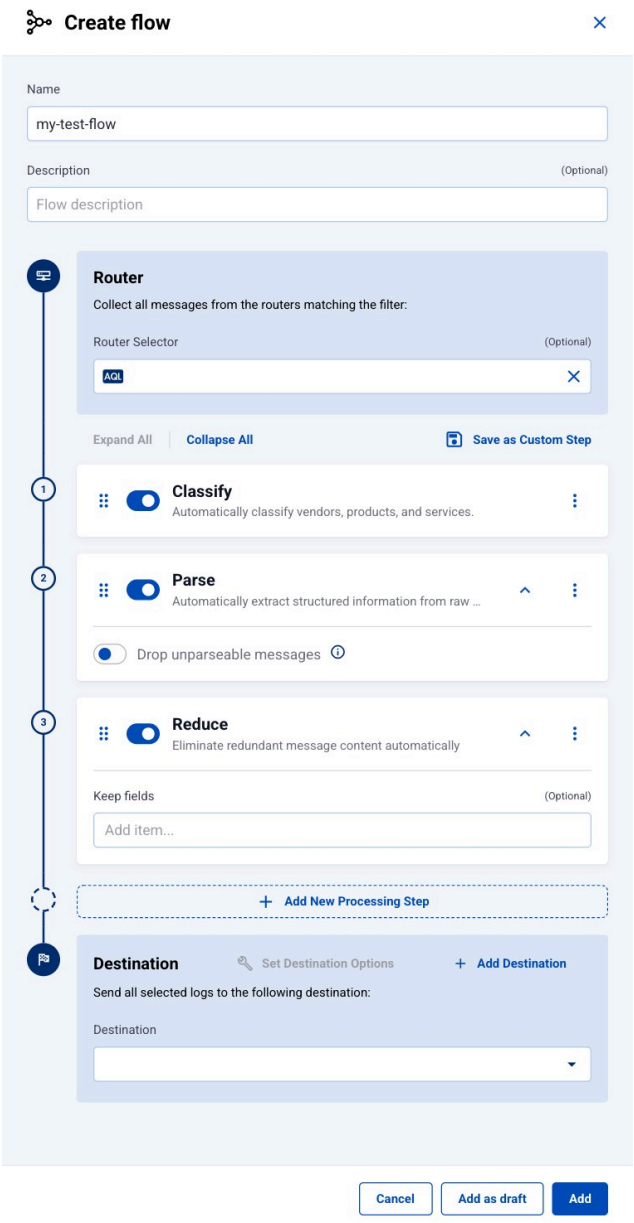
5. Select the **Destination** where you want to send your data. If you don't have any destination configured, you can select **Add Destination** in the destination section to create a new destination now. For details on the different destinations, see Destinations.

   - If you don't have any destination configured, see Destinations.
   - If you've already created a store, it's automatically available as a destination. Note that the **Router Selector** of the flow must match only AxoRouters that have the selected store available, otherwise you'll get an error message.
   - If you want to send data to another AxoRouter, enable the **Show all destinations** option, and select the connector of the AxoRouter where you want to send the data.



6. (Optional) To process the data transferred in the flow, select **Add New Processing Step**. For details, see Processing steps. For example:

1. Add a **Classify**, a **Parse**, and a **Reduce** step, in that order, to automatically remove redundant and empty fields from your data.
2. To select which messages are processed by the flow, add a **Select Messages** step, and enter a filter into the **AQL Expression** field. For example, to select only the messages received from Fortinet FortiGate firewalls, use the `meta.vendor = fortinet AND meta.product = fortigate` query.



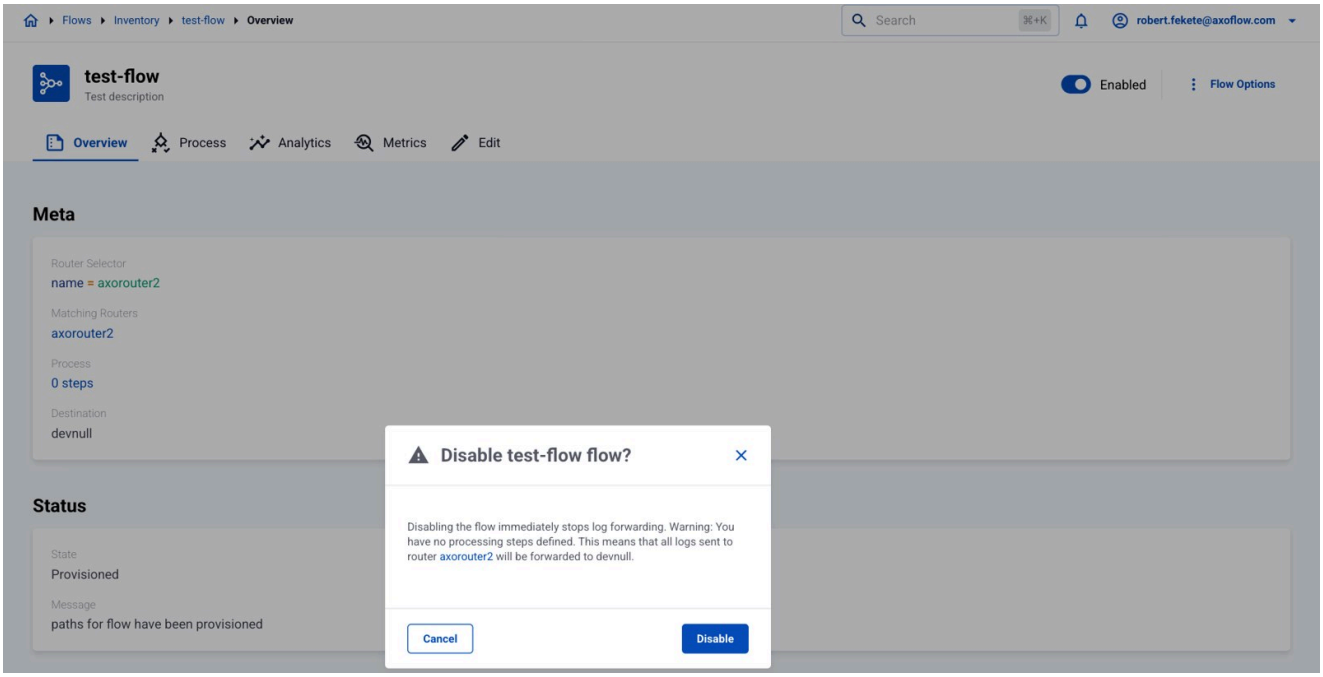
7. Select **Add**, then confirm with **Add**. To save the flow without provisioning it to the routers yet, select **Add as draft** instead.
8. The new flow appears in the **Flows** list.



Disable flow

You can disable a flow without deleting it: select the toggle next to the flow name in the **Flows** list, or the **Enabled** toggle on the page of the flow, then select **Disable** to confirm.

> **CAUTION:**
>
> Disabling a flow immediately stops log forwarding for the flow. Any data that's not forwarded using another flow can be irrevocably lost.



10.3 Processing steps

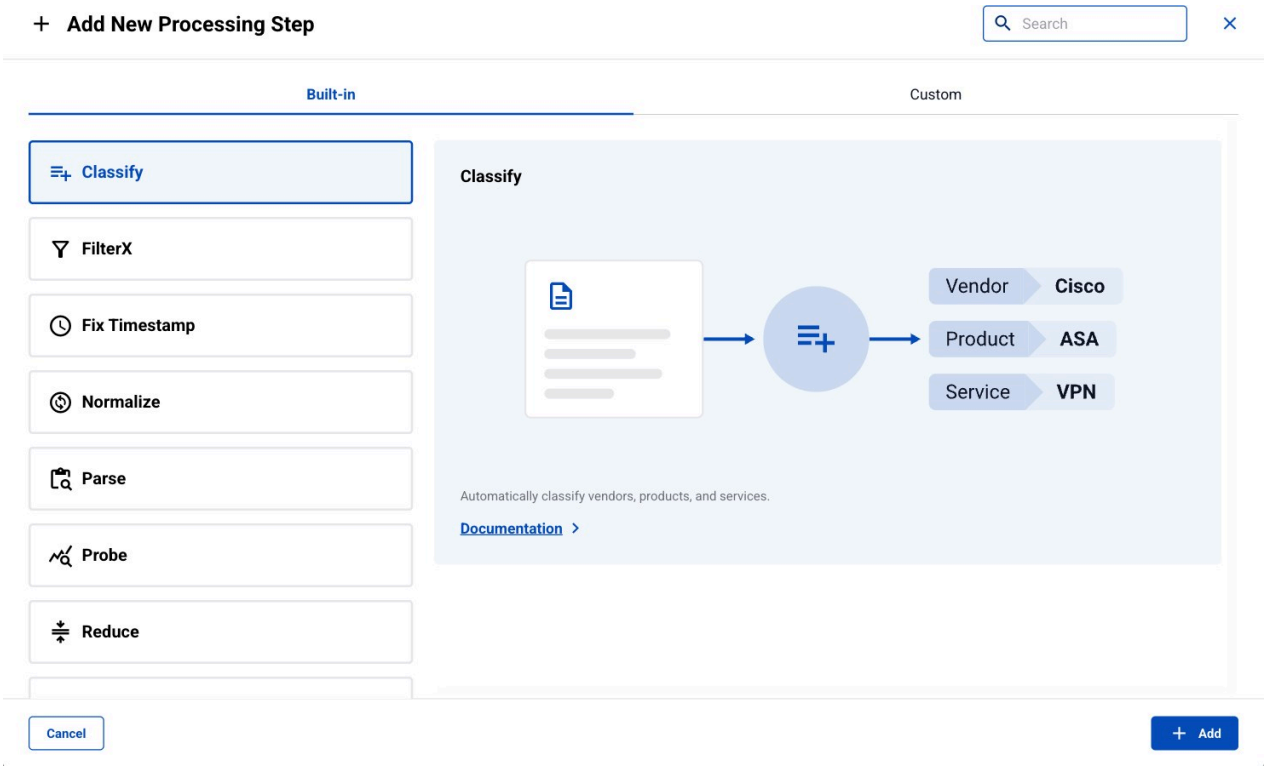
Add and configure processing steps in Axoflow flows to classify, parse, reduce, normalize, and filter data, and reuse them as custom steps.

Flows can include a list of *processing steps* to select and transform the messages. The AxoRouters of the flow send all the log messages they receive through the processing steps. Then they forward the filtered and processed messages to the destination of the flow.

> **CAUTION:**
>
> AxoRouter runs the processing steps in the order they appear in the flow. Make sure to put the steps in the correct order.

To add processing steps to an existing flow, complete the following steps.

1. Open the **Flows** page and select the flow you want to modify. Alternatively, select `⌘/Ctrl + K` and enter the name of the flow.
2. Select **Process > Add New Processing Step**.
3. Select the type of the processing step to add.



The following types of processing steps are available:

- **Classify**: Automatically classify the data and assign vendor, product, and service labels. This step (or its equivalent in the Connector rule) is required for the **Parse** and **Reduce** steps to work.

- **FilterX**: Execute an arbitrary FilterX snippet.
- **Fix Timestamp**: Correct timestamps that have a wrong or missing timezone, or replace them with the time AxoRouter received the message.
- **Normalize**: Normalize the data according to the specified standard: Azure Monitor, Cortex Data Model (XDM), CrowdStrike Parsing Standard (**CrowdStrike**), Dynatrace, Elastic Common Schema (**ECS**), or Open Cybersecurity Schema Framework (**OCSF**).
- **Parse**: Automatically parse the data to extract structured information from the content of the message. The **Classify** step (or its equivalent in the Connector rule) is required for the **Parse** step to work.
- **Probe**: A measurement point that provides metrics about the throughput of the flow.
- **Reduce**: Automatically remove redundant content from the messages. The **Classify** and **Parse** steps (or their equivalent in the Connector rule) are required for the **Reduce** step to work.
- **Regex**: Execute a regular expression on the message.
- **Select Messages**: Filter the messages using a query. Only the matching messages go to the next steps.
- **Set Fields**: Set the value of one or more fields on the message object.
- **Unset Field**: Unset the specified field of the message object.

To add a custom step that you've created earlier, select the **Custom** tab, then select the custom step.

4. Configure the processing step as needed for your environment.

5. (Optional) To apply the processing step only to specific messages from the flow, add a condition to the step: select more_vert > **Add Condition** on the step, enter the condition, then select **Save**. The condition appears in the header of the step. Conditions act as filters, but apply only to this step. The step doesn't change the messages that don't match the condition. These messages go to the next step. For details about the message schema and the available fields, see Message schema reference.

   Conditions use AQL queries. You can make complex filters with the Equals, Contains (partial match), and Match (regular expression match) operators. Note that:

   - AxoConsole autocompletes the built-in and custom labels and field names, as well as their most frequent values, but doesn't autocomplete labels and variables created by data parsing and processing steps. See the Message schema for details.
   - You can use the `AND` and `OR` operators to combine expressions, and also parentheses if needed.

6. (Optional) To describe the purpose of the step, select more_vert > **Add Description** on the step, enter the description, then select **Save**.
7. (Optional) To change the location of the step in the flow, drag the step to a new location.

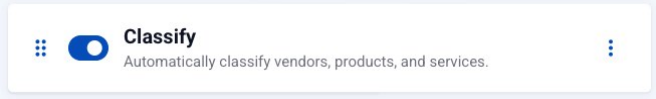
   The more_vert menu of the step also has the following actions:

   - **Duplicate**: Insert a copy of the step after the step.
   - **Add Step Before** and **Add Step After**: Insert a new processing step before or after the step.
   - **Delete**: Remove the step from the flow.

   To temporarily turn off a step without removing it, use the toggle in the header of the step.
8. Select **Save**.

Classify

Automatically classify the data and assign vendor, product, and service labels. This step (or its equivalent in the Connector rule) is a prerequisite for the **Parse** and **Reduce** steps, and must be executed before them. AxoRouter can parse data from the supported data sources. If your source isn't listed, contact us.

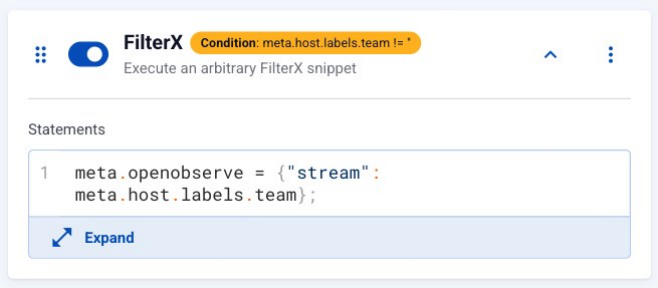


FilterX

Execute an arbitrary FilterX snippet. The following example sets the `meta.openobserve` variable to a JSON object. The `stream` key of the object gets the value of the `meta.host.labels.team` field. The step uses a condition, so it runs only on messages where the `meta.host.labels.team` field isn't empty.

- **Condition** (set in the **Add Condition** dialog of the step): `meta.host.labels.team != ''`
- **Statements**:

```
meta.openobserve = {"stream": meta.host.labels.team};
```



> **Note**
>
> **To parse a log message that AxoRouter can't parse automatically, you can use ai-assisted parsing to create a FilterX processing step.**

Fix timestamp

Correct the timestamp of messages that have a wrong or missing timezone. For example, many devices send their local time without a timezone, so AxoRouter interprets the timestamp in the wrong timezone, and the message appears hours earlier or later than it happened.

Prerequisite: The **Fix Timestamp** step requires AxoRouter version 0.78.0 or newer.

Set the following options:

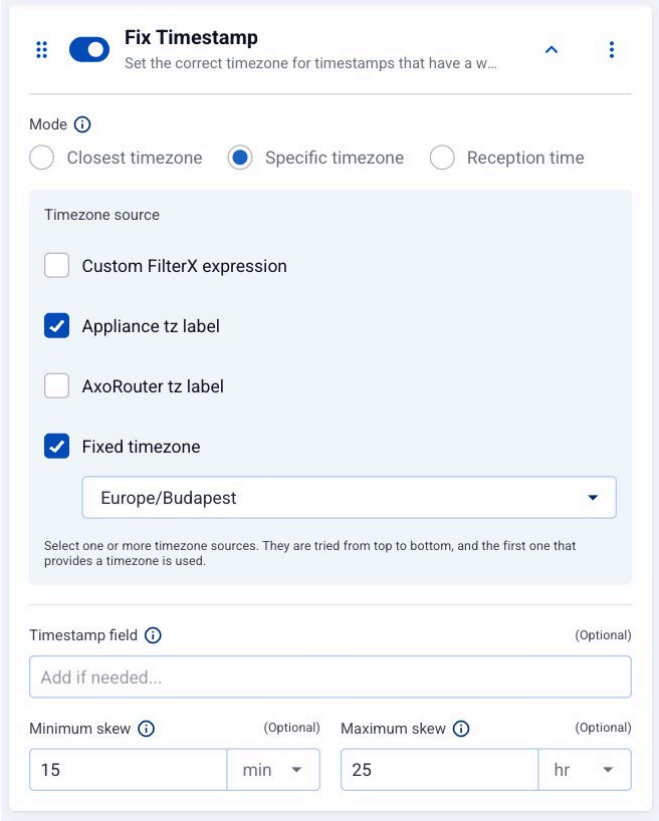
- **Mode**: How to correct the timestamp.
  - **Closest timezone** (default): Use the timezone that brings the timestamp closest to the time AxoRouter received the message. The step keeps the date and time of the timestamp, and changes only its timezone. If no timezone fits, the timestamp stays unchanged. The step doesn't change rehydrated messages in this mode, because their reception time is the time of the rehydration.
  - **Specific timezone**: Use the timezone of the **Timezone source** below. The step keeps the date and time of the timestamp, and changes only its timezone.
  - **Reception time**: Replace the timestamp with the time AxoRouter received the message. Use this mode if the timestamp of the message is unreliable. This mode also sets the timestamp of messages that don't have one.
- **Timezone source** (only for the **Specific timezone** mode): Select one or more sources. AxoRouter tries them from top to bottom, and uses the first one that provides a timezone. If none of them does, the timestamp stays unchanged.
  - **Custom FilterX expression**: A FilterX expression that returns the name of a timezone, for example, `meta.host.labels.tz`.

  - **Appliance tz label**: The value of the `tz` label of the source host (appliance) that sent the message, for example, `Europe/Budapest`. For details on adding labels to hosts, see Custom labels and metadata. This source requires the **Classify** step (or its equivalent in the Connector rule) before the **Fix Timestamp** step.
  - **AxoRouter tz label**: The value of the `tz` label of the AxoRouter that processes the message. AxoRouters without a `tz` label skip this source.
  - **Fixed timezone**: A timezone name, for example, `Europe/Budapest`. Numeric offsets like `+02:00` don't work. To set a fixed offset, use the `Etc/GMT` timezone names, but note that their sign is inverted: for UTC+2, use `Etc/GMT-2`. Use this source last, as a fallback for the other sources.
- **Timestamp field**: The field that contains the timestamp to correct. Default value: `log.time_unix_nano`. For details on the available fields, see Message schema reference.
- **Minimum skew** and **Maximum skew**: Correct the timestamp only if it differs from the reception time by at least the minimum, and at most the maximum skew. Leave them empty for no limit. For example:
  - A **Minimum skew** of `15m` skips differences that no timezone can explain, because the smallest timezone offset in use is a quarter of an hour.
  - A **Maximum skew** of `25h` skips differences that are too large to be a timezone offset, because the largest timezone offset in use is 14 hours.

  The **Minimum skew** can't be larger than the **Maximum skew**. In **Reception time** mode, the step always sets the timestamp of messages that don't have one, regardless of the skew limits.

The following example sets the timezone of messages from the `tz` label of the source host. If the host has no such label, it uses the `Europe/Budapest` timezone. The step changes only timestamps that are 15 minutes to 25 hours off from the reception time.

- **Mode**: **Specific timezone**
- **Timezone source**: **Appliance tz label**, **Fixed timezone** (`Europe/Budapest`)
- **Minimum skew**: `15m`
- **Maximum skew**: `25h`

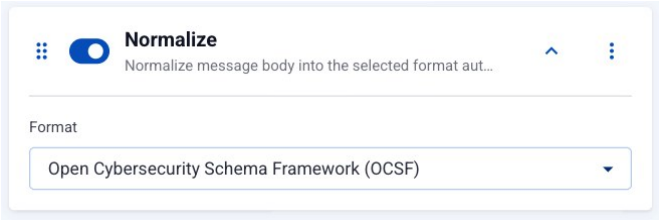


Normalize

Prerequisite: It's recommended to enable the **Classify** step (or its equivalent in the Connector rule) when using the **Normalize** step. Normalization implicitly performs the **Parse** step as well.

The **Normalize** step reformats the content of the message to the selected normalization scheme. AxoRouter supports the following normalization schemas:

- Azure Monitor
- Cortex XSIAM (XDM)
- CrowdStrike Parsing Standard (CPS) (**CrowdStrike**)
- Dynatrace
- Elastic Common Schema (**ECS**)
- Open Cybersecurity Schema Framework (**OCSF**)



Parse

Prerequisite: The **Classify** step (or its equivalent in the Connector rule) is required for the **Parse** step to work.

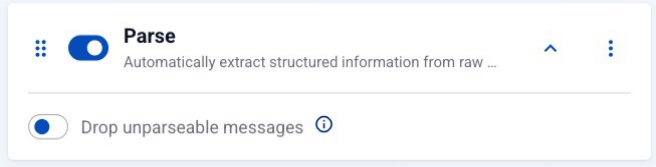
The **Parse** step automatically parses the data from the content of the message, and replaces the message content (the `log.body` field in the internal message schema) with the structured information. AxoRouter can parse data from the supported data sources. If your source isn't listed, contact us. Alternatively, you can create your own parser using **FilterX** processing steps.

For example, if the message content is in the Common Event Format (CEF), AxoRouter creates a structured FilterX dictionary from its content, like this:

```
CEF:0|KasperskyLab|SecurityCenter|13.2.0.1511|KLPRCI_TaskState|Completed
successfully|1|foo=foo bar=bar baz=test
```

Parsed and structured content:

```
{
"version":"0",
"device_vendor":"KasperskyLab",
"device_product":"SecurityCenter",
"device_version":"13.2.0.1511",
"device_event_class_id":"KLPRCI_TaskState",
"name":"Completed successfully",
"agent_severity":"1",
"foo":"foo",
"bar":"bar",
"baz":"test"
}
```



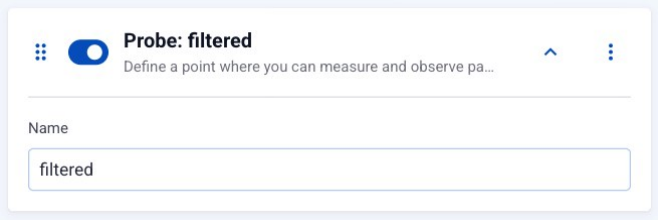
By default, AxoRouter forwards the messages that it can't parse to the next processing step of the flow. If you want to discard such messages instead, enable the **Drop unparseable messages** option.

> **Note**
>
> **To parse a log message that AxoRouter can't parse automatically, you can use ai-assisted parsing.**

Probe

Set a measurement point that provides metrics about the throughput of the flow. Note that:

- You can't name your probes `input` and `output`, as these are reserved.
- Probe names can contain only lowercase alphanumeric characters, `-`, and `.`, and must start and end with an alphanumeric character.
- If you add a probe called `filtered` to the flow, AxoRouter uses this point as the input metric data point to calculate the data reduction of the flow in the **Dashboard** page.



For details, see Flow metrics.

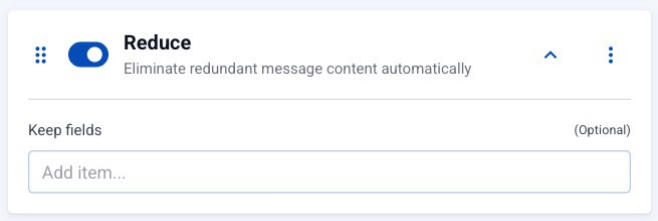
Reduce

Prerequisite: The **Classify** and **Parse** steps (or their equivalent in the Connector rule) are required for the **Reduce** step to work.

The **Reduce** step automatically removes redundant content from the messages.

- If you want to keep certain fields unchanged, use the **Keep fields** field. For details on the message fields, see Message schema.
- To see what changed in a message, add a **Probe** processing step before and after the reduce step, then use Flow tapping.

For details on how data reduction works, see Classify and reduce security data.



Regex

Use a regular expression to process the message.

- **Input** is an AxoSyslog template string, for example, `${MESSAGE}`. AxoRouter matches the regular expression with this string.
- **Pattern** is a Perl Compatible Regular Expression (PCRE). If you want to match the pattern to the whole string, use the `^` and `$` anchors at the beginning and end of the pattern.

  You can use https://regex101.com/ with the ECMAScript flavor to validate your regex patterns. (For named capture groups, use the `(?<group_name>pattern)` syntax. The `(?P<group_name>pattern)` syntax doesn't work.)

Named or anonymous capture groups show the parts of the pattern that AxoRouter extracts to the **Field** field. Specify a named capture group like `(?<group_name>pattern)`. Its value is available as `field_name["group_name"]`. AxoRouter numbers the anonymous capture groups. For example, the value of the second group from `(first.*) (bar|BAR)` is available as `field_name["2"]`. When specifying variables, note the following points:

- If you want to include the results in the outgoing message or in its metadata for other processing steps, you must:
  - Reference an existing variable or field. AxoRouter updates this field with the matches.
  - For new variables, use names for the variables that start with the `$` character, or
  - declare them explicitly. For details, see the FilterX data model in the AxoSyslog documentation.
- If the pattern doesn't match, AxoRouter sets the field to an empty dictionary (`{}`). For details about the message schema and the available fields, see Message schema reference.

Regex example: Parse the raw message

The following processing step parses an unclassified log message from a firewall into a structured format in the `log.body` field:

- **Input**: `${RAWMSG}`
- **Pattern**:

```
^(?<priority><\d+>)(?<version>\d+)\s(?<timestamp>[\d\-T:.Z]+)\s(?<hostname>
[\w\-.]+)\s(?<appname>[\w\-]+)\s(?<event_id>\d+|\-)\s(?<msgid>\-)\s(?
<unused>\-)\s(?<rule_id>[\w\d]+)\s(?<network>\w+)\s(?<action>\w+)\s(?
<result>\w+)\s(?<rule_seq>\d+)\s(?<direction>\w+)\s(?<length>\d+)\s(?
<protocol>\w+)\s(?<src_ip>\d+\.\d+\.\d+\.\d+)\/(?<src_port>\d+)->(?
<dest_ip>\d+\.\d+\.\d+\.\d+)\/(?<dest_port>\d+)\s(?<tcp_flags>\w+)\s(?<policy>
[\w\-.]+)
```

- **Field**: `log.body`



Regex example: Parse an existing field

The following processing step processes the entire `log.body` field for messages generated by the AxoSyslog `loggen` command and stores it in the `log.pad` field, for example:

```
2024-05-02T10:56:31.000000+00:00 localhost prg00000[1234]: seq: 0000000065, thread:
0000, runid: 1714647391, stamp: 2024-05-02T10:56:31 PADDPADDPADDPADD
```

- **Input**: `log.body`
- **Pattern**:

```
seq:\s+(?<seq>\d+),\s+thread:\s+(?<thread>\d+),\s+runid:\s+(?
<runid>\d+),\s+stamp:\s+(?<stamp>[^\s]+)
```

- **Field**: `log.pad`

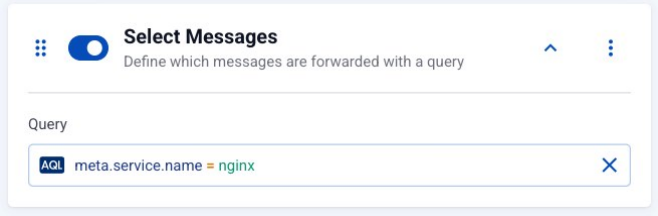
For details about the message schema and the available fields, see Message schema reference.

Select messages

Filter the messages using a query. Only the matching messages go to the next steps.

The following example selects messages that have the `meta.service.name` label set to `nginx`.

- **Query**: `meta.service.name = nginx`



You can also select messages using various other metadata about the connection and the host that sent the data, the connector that received the data, the classification results, and also custom labels, for example:

- a specific connector: `meta.connector.name = axorouter-mysyslog-connector`
- a type of connector: `meta.connector.type = otlp`
- a sender IP address: `meta.connection.src_ip = 192.168.1.1`
- a specific product, if classification is enabled in an earlier processing step or in the connector that received the message: `meta.product = fortigate` (see Vendors for the metadata of a particular product)
- a custom label you've added to the host: `meta.host.labels.location = eu-west-1`

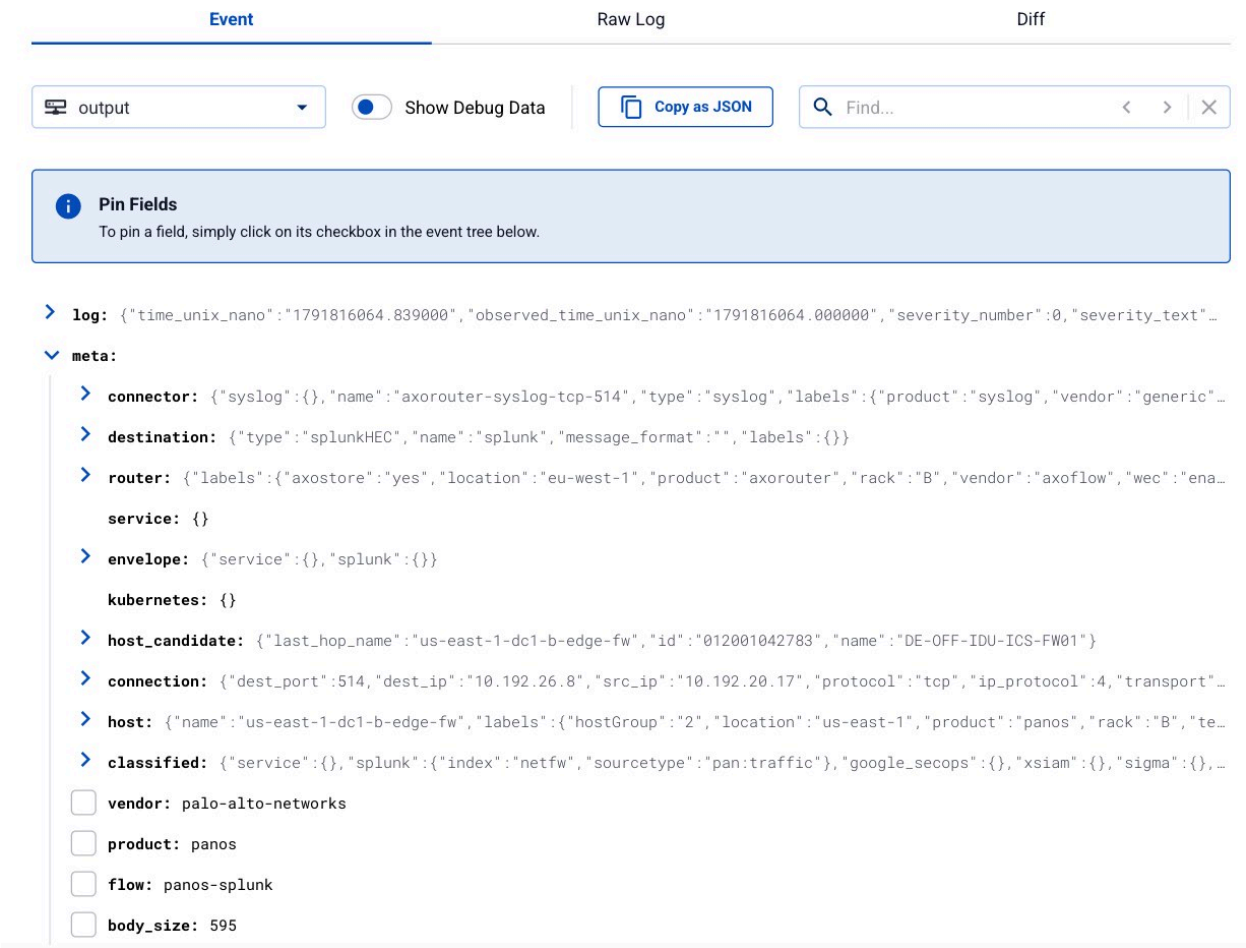
In addition to exact matches, you can use the following operators:

- `!=` not equal (string match)
- `=*` contains (substring match)
- `!*` : doesn't contain
- `=~` : matches regular expression
- `!~` : doesn't match regular expression
- `==~` : matches case sensitive regular expression
- `!=~` : doesn't match case sensitive regular expression

You can combine multiple expressions using the AND (`+`) operator. (To delete an expression, press SHIFT+Backspace.)

Note

To check the metadata of a particular message, you can use flow tapping or Log tapping. In the **Event Details** of a message, the metadata is under the `meta` key of the **Event** tab.



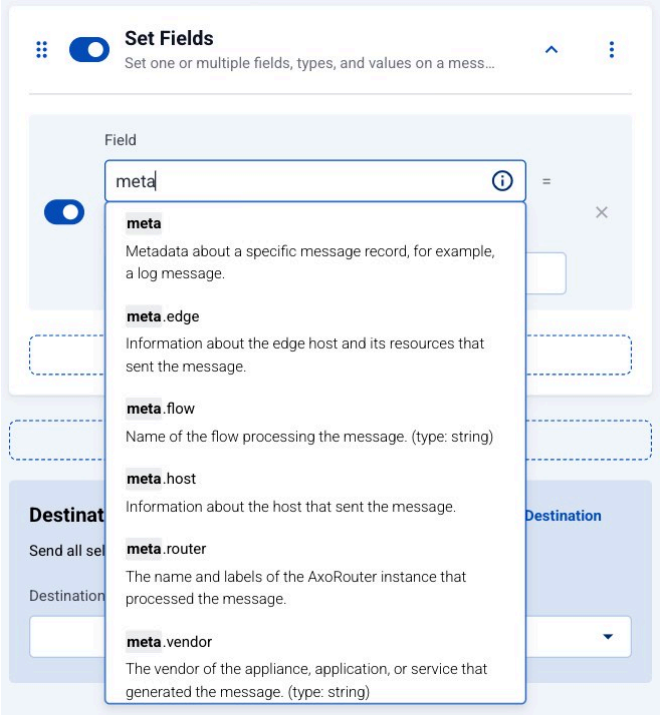
For details about the message schema and the available fields, see Message schema reference.

Set fields

Set specific field of the message. You can use static values, and also dynamic values that were extracted from the message automatically by Axoflow or manually in a previous processing step. If the field doesn't exist, it's automatically created.

The type of the field can be `boolean`, `string`, `number`, and `expression`. Select **expression** to set the value of the field using a FilterX expression.

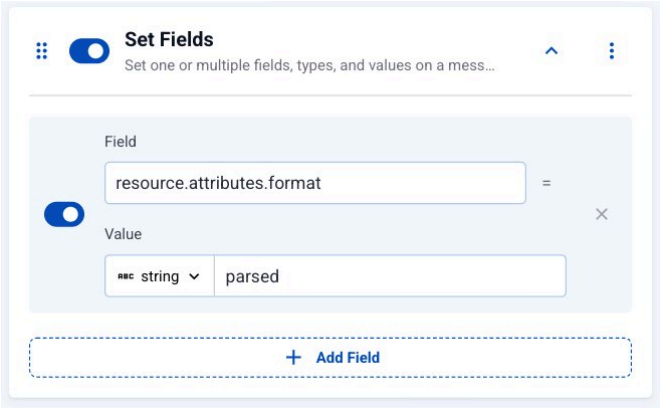
If you start to type the name of the field, AxoConsole shows matching and similar field names based on the message schema:



To set more fields in the same step, select **Add Field**. To temporarily skip a field without removing it, use the toggle next to the field.

The following example sets the `resource.attributes.format` field to `parsed`.

- **Field**: `resource.attributes.format`
- **Value**: `string`, `parsed`



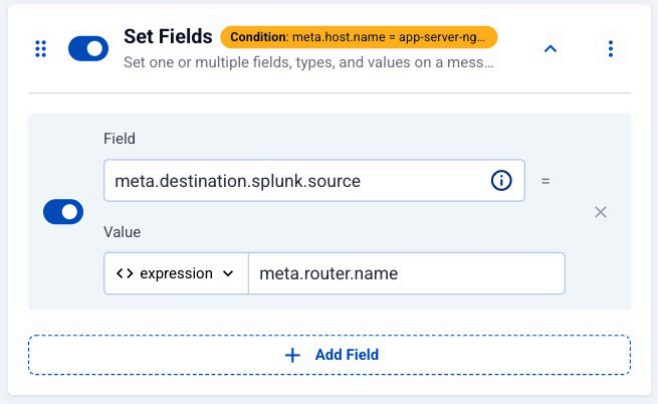
Set the Splunk source field

The following example sets the Splunk source field to the name of the AxoRouter host. Add a condition to the step (more_vert > **Add Condition**) to match only the hosts where you want to set the source field. If you don't set a condition, the step sets the source field of every message.

- **Condition** (set in the **Add Condition** dialog of the step): `meta.host.name = app-server-nginx`

- **Field**: `meta.destination.splunk.source`
- **Value**: `expression` , `meta.router.name`

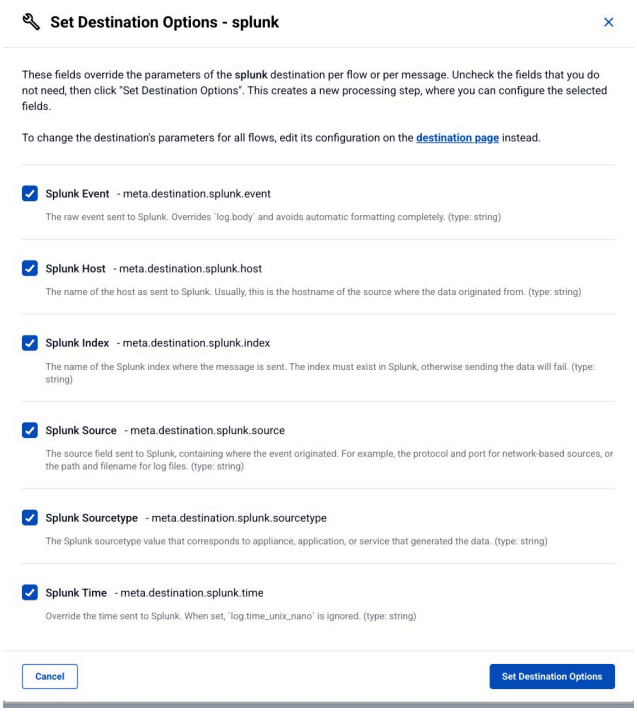
> **Note**
>
> **If the Axoflow classification doesn't set the source field for the message automatically, and you haven't set it in a flow processing step manually (by setting the `meta.destination.splunk.source` field), AxoRouter automatically sets the source to the name of the AxoRouter connector that received the message (for example, `axorouter-syslog-tcp-514`).**



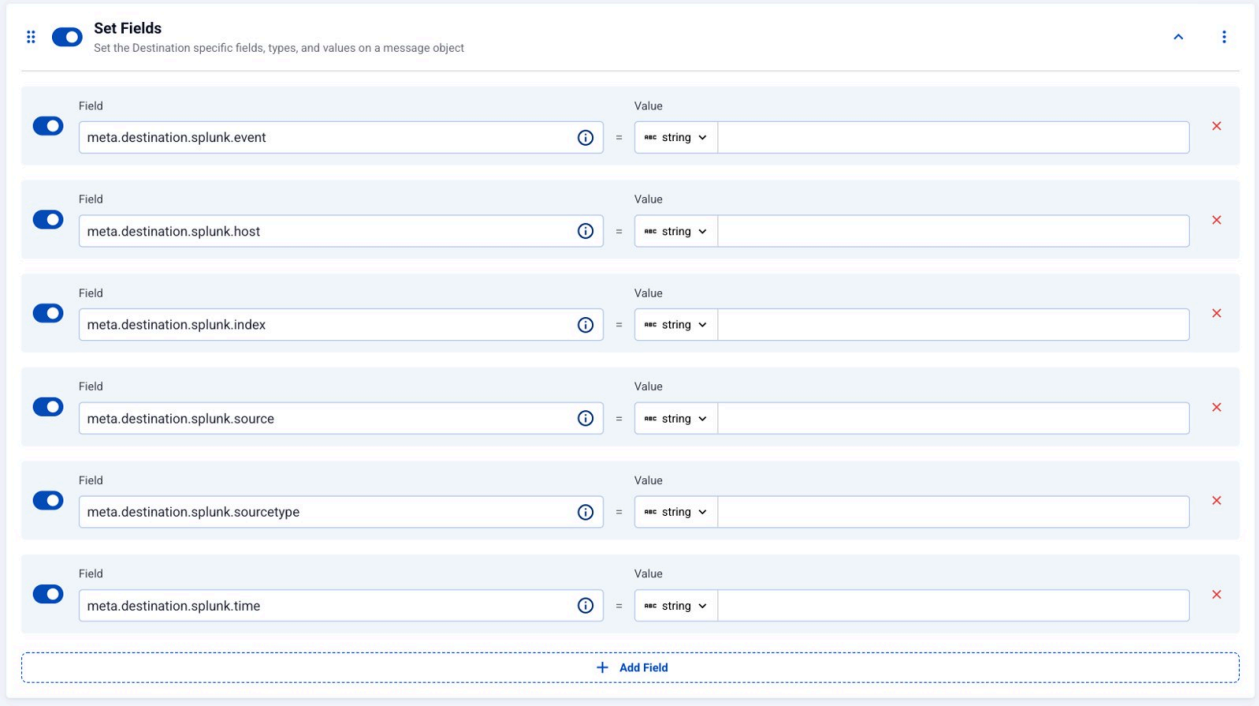
Set destination options

Destination options are destination-specific fields that override the parameters of the destination for a flow or for specific messages, for example, the index for a Splunk destination, or the stream name for an OpenObserve destination. To change the parameters of the destination for every flow, edit the destination instead.

1. In the **Destination** section of the flow, select **Set Destination Options**. The **Set Destination Options** dialog lists the fields that the destination supports.



2. Clear the fields that you don't need, then select **Set Destination Options**. AxoConsole inserts a **Set Fields** processing step into the flow with the selected fields.



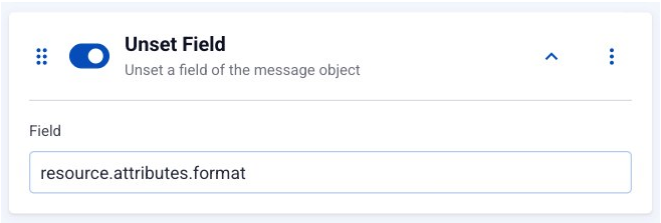
3. Set the values of the fields.

You can insert this processing step multiple times if you want to set the fields to different values using conditions.

When using **Set Destination Options**, note that if you leave a parameter empty, AxoRouter sends it as an empty field to the destination. If you don't want to send that field at all, delete the field.

Unset field

Unset a specific field of the message.

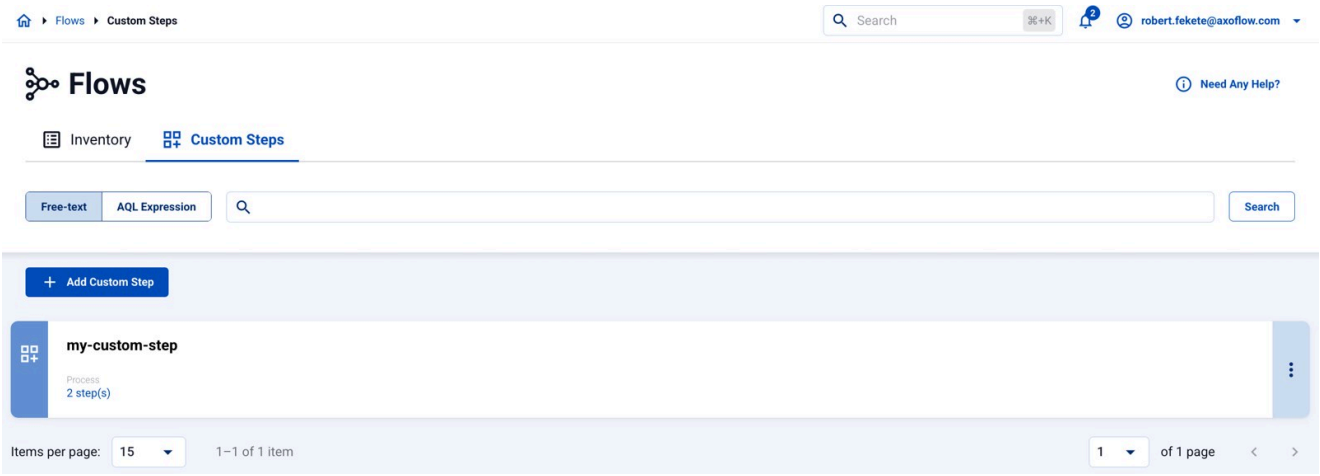


Custom steps

A *custom step* is a named, reusable sequence of processing steps. Create it once, then add it to multiple flows, connector rules, or other custom steps like any built-in processing step. For example, you can build a parsing and data reduction pipeline for a specific data source once, and reuse it everywhere you process that data.

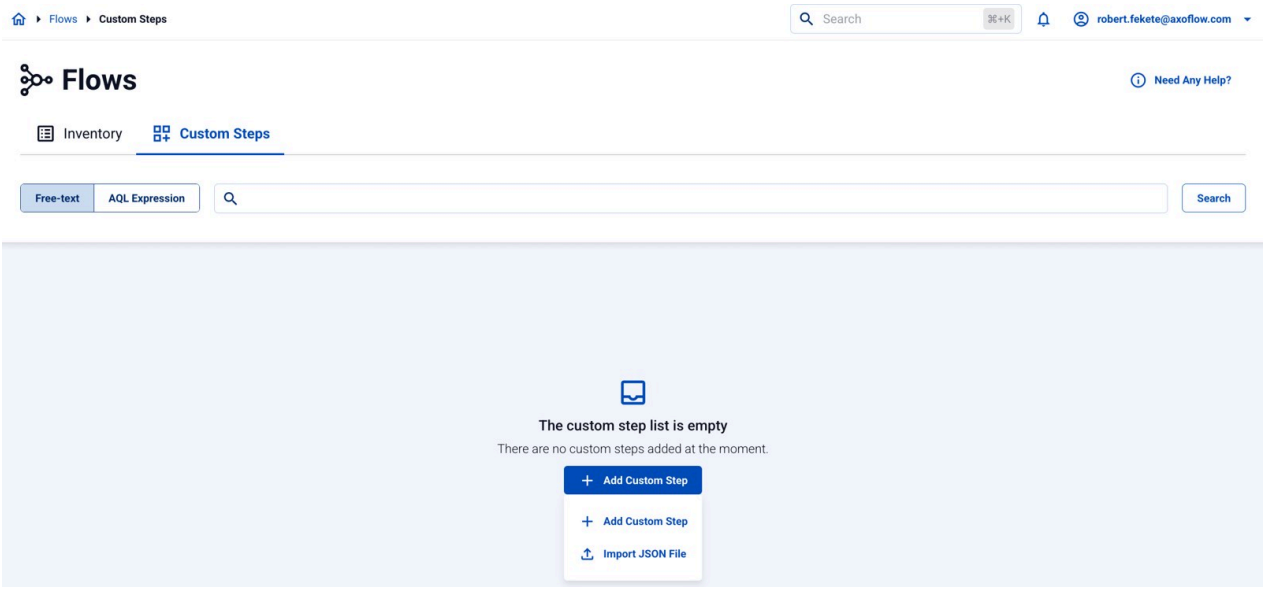
A custom step is a reference: when you modify a custom step, the change applies to every flow, connector rule, and custom step that uses it.

To manage your custom steps, open **Flows > Custom Steps**. For each custom step, the list shows how many processing steps it contains, and where it's used. Select the usage links to see the flows, connector rules, and other custom steps that include the custom step.

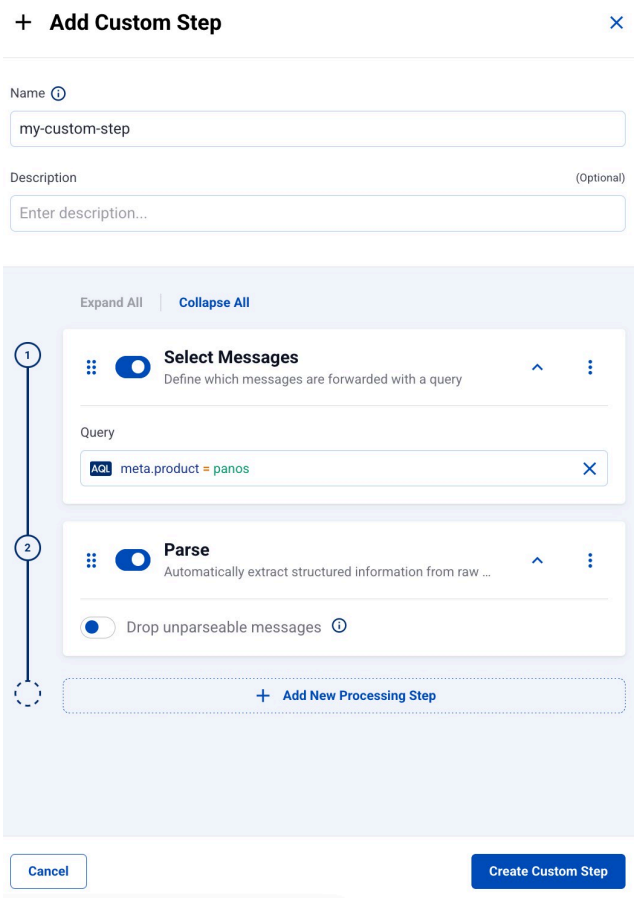


Create a custom step

1. Open **Flows > Custom Steps**, then select **Add Custom Step > Add Custom Step**.



2. Set the **Name** of the custom step. This name appears in the processing step list when you add the custom step to a flow. Note that:
   - The name can contain only lowercase alphanumeric characters, `-`, and `.`, and must start and end with an alphanumeric character.
   - You can't change the name later.
3. (Optional) Enter a **Description** for the custom step.
4. Add the processing steps of the custom step. A custom step can contain other custom steps, but it can't contain **Probe** steps.



5. Select **Create Custom Step**.

You can also create a custom step from the processing steps you've already configured:

- **Save existing steps**: When editing the processing steps of a flow, a connector rule, or a rehydration, select **Save as Custom Step**. AxoConsole creates a new custom step that contains the current processing steps. Set a name for the custom step, then select **Create Custom Step**.
- **Import from a file**: Select **Add Custom Step > Import JSON File**, and upload a custom step that you've exported earlier.
- **Clone**: In the list of custom steps, open the menu of a custom step and select **Clone**.

Modify a custom step

1. Open **Flows > Custom Steps**, then select the custom step.
2. Modify the custom step:

> **CAUTION:**
>
> Changes to a custom step affect every flow, connector rule, and custom step that uses it. Before modifying a custom step, check where it's used.

- To change the processing steps, select **Process**, edit the steps, then select **Save**.
- To change the description, select **Edit**, modify the description, then select **Save**.

Alternatively, when a flow includes a custom step, select more_vert > **Navigate to custom step** on the custom step in the flow to open the custom step in a new tab.

Nested custom steps

A custom step can include other custom steps. However, you can't create reference loops:

- A custom step can't include itself.
- A custom step can't include another custom step that includes the first custom step, either directly or through other custom steps. For example, if `step-a` includes `step-b`, then `step-b` can't include `step-a`.

AxoConsole rejects changes that would create a loop, and also rejects references to custom steps that don't exist.

Export custom steps

To export a custom step into a JSON file, open **Flows > Custom Steps**, open the menu of the custom step, then select **Export As JSON File**. You can import the file into another Axoflow deployment.

Delete custom steps

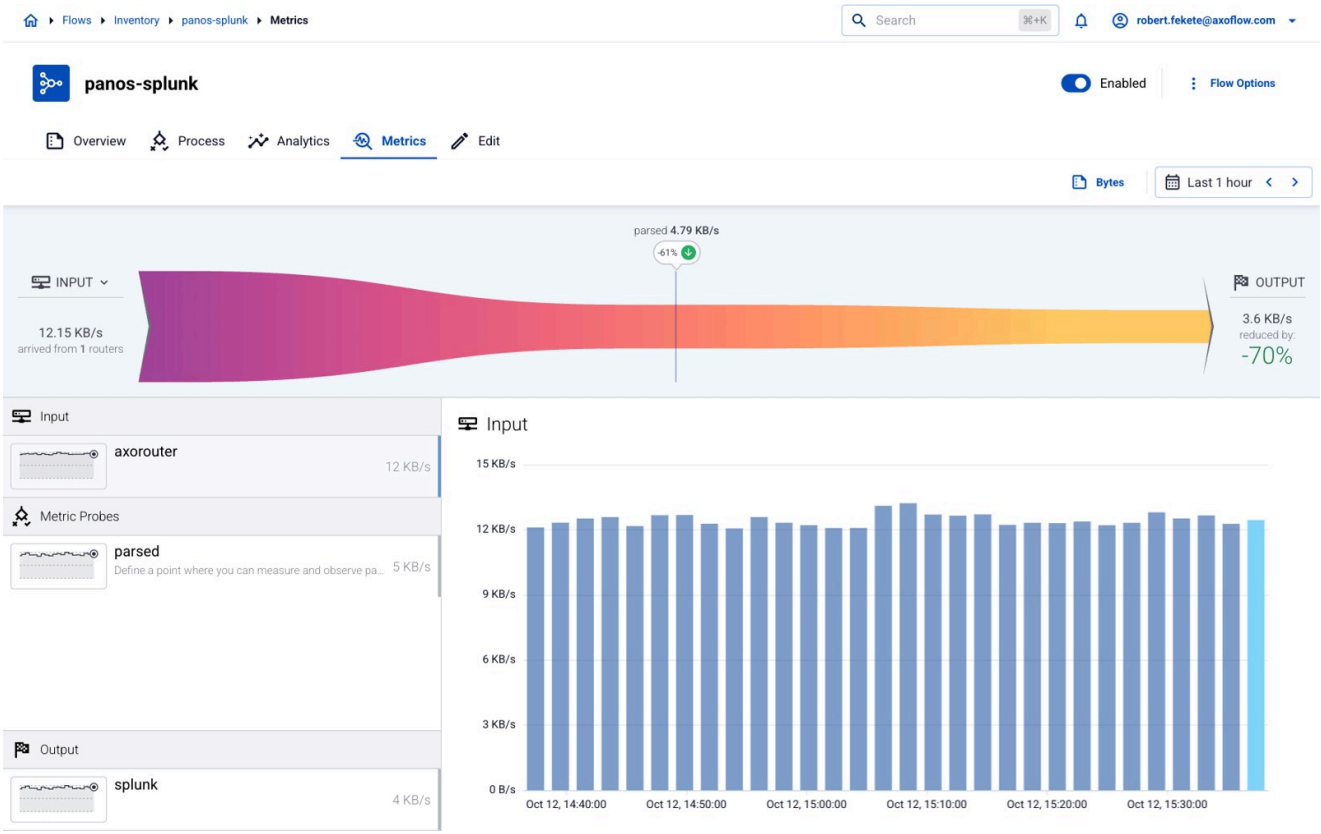
To delete a custom step, open **Flows > Custom Steps**, open the menu of the custom step, then select **Delete**. You can't delete a custom step that's used by a flow, a connector rule, or another custom step. Remove the custom step from these resources first.

10.4 Flow metrics

Check the events and bytes per second at the input, output, and probe steps of a flow on its funnel and time-series charts.

The **Metrics** page of a flow shows the amount of data (in events per second or bytes per second) processed by the flow. By default, the flow provides metrics for the input and the output of the flow. To get more information about the data flow, add **Probe** steps to the flow. For example, you can add a probe after:

- **Select Messages** steps to check the amount of data that match the filter, or
- **Reduce** steps to check the ratio of data saved.

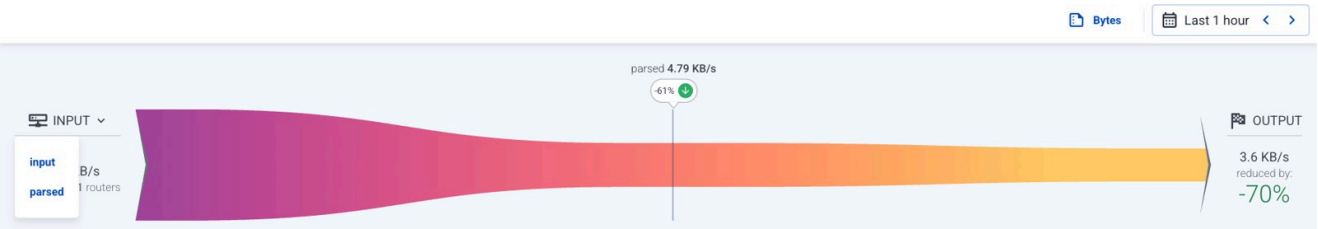


Note

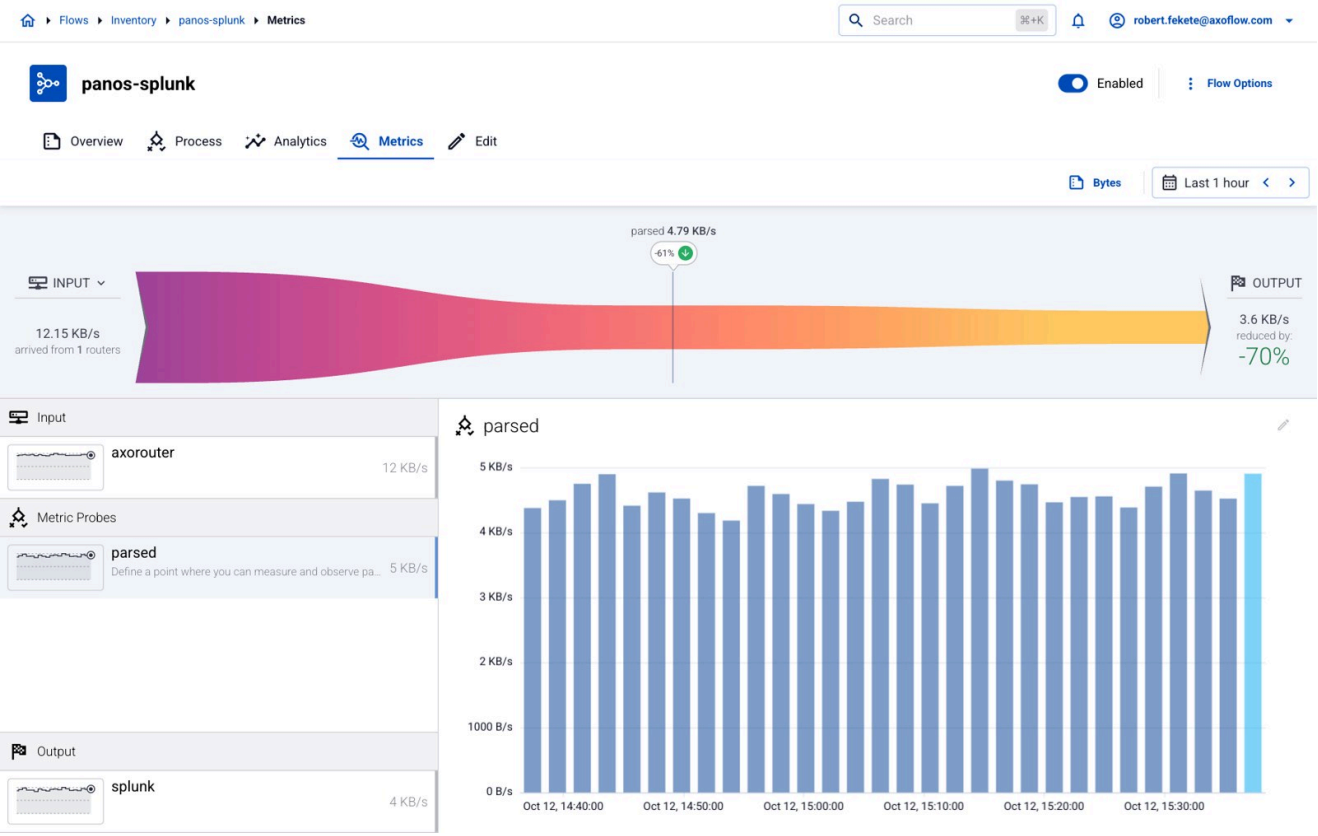
By default, Axoflow stores metrics for 30 days, unless specified otherwise in your support contract. Contact us if you want to increase the data retention time.

The top of the page is a funnel graph that shows the amount of data (in events/sec or bytes/sec) processed at each probe of the flow. To show the funnel graph between a probe and the output, click the **INPUT** field on the left of the funnel graph and select a probe. That way you can display a diagram that starts at a probe (for example, after the initial Select Messages step of the flow) and shows all subsequent probes. Note that

OUTPUT on the right of the funnel graph shows the relative amount of data at the output compared to the input. This value refers to the entire flow, it's not affected by selecting a probe for the funnel graph.



The bottom part of the page shows how the amount of data processed by a probe (by default, the input probe) changed over time. Select a probe in the **Metric Probes** list on the left (or **Input** or **Output**) to show its metrics. To edit the probe, select edit (**Edit probe**) above its chart: it opens the processing steps of the flow. Click on the timeline to show the input-probe-output states corresponding to the selected timestamp. Clicking the timeline updates the funnel graph as well. You can adjust the time range in the filter bar at the top.



You can adjust the data displayed using the filter bar:

- **Time Range**: Select the calendar_month icon to change the time period that's displayed on the charts. You can use absolute (calendar) time, or relative time (for example, the last 2 days). To move to the previous or the next period of the same length, select the arrows next to the time period (**Previous interval** and **Next interval**).

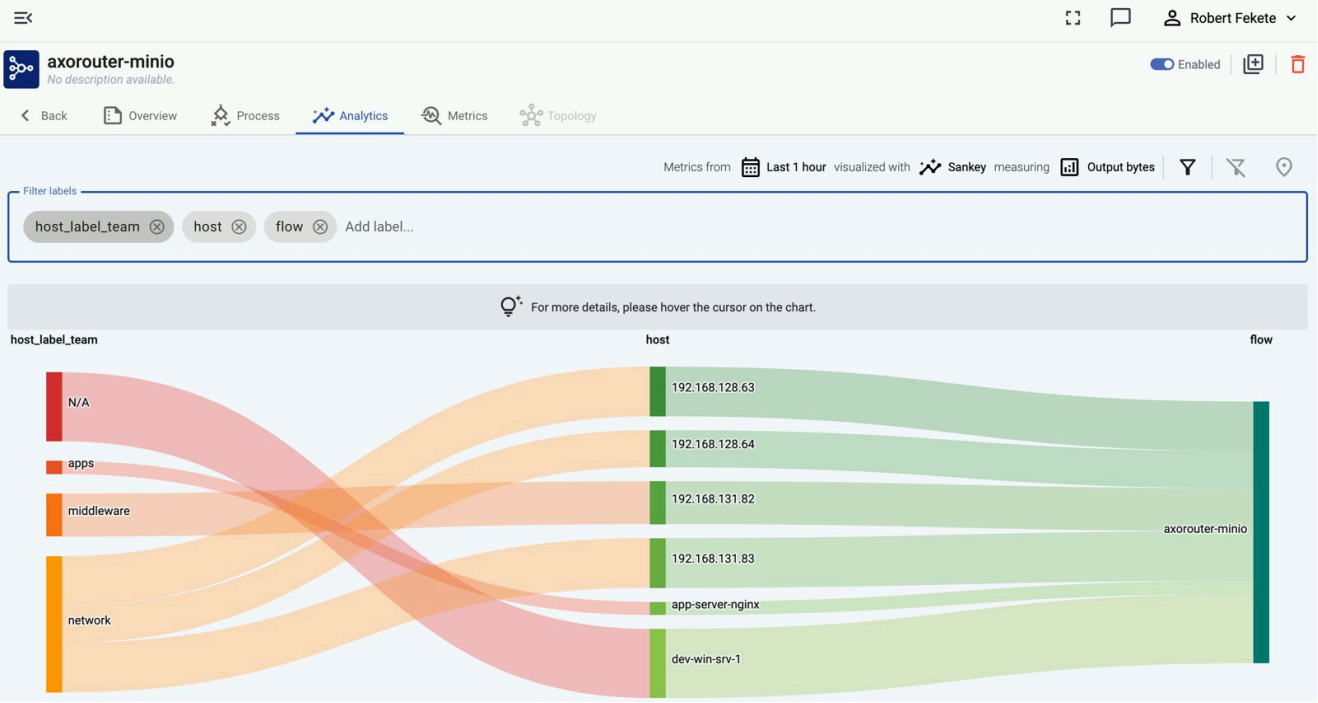
Axoflow stores all dates in Coordinated Universal Time (UTC), and automatically converts it to the timezone of set in your browser/operating system.

- Data unit: Select the unit button (summarize **Bytes** or **Events**) to switch the charts between bytes per second and events per second.
- The settings of the filter bar change the URL parameters of the page, so you can bookmark it, or share a specific view by sharing the URL.

10.5 Flow analytics

Analyze the throughput of a single flow with Sankey and Sunburst diagrams on its Analytics page.

The **Analytics** page of a flow allows you to analyze the data throughput of the flow using Sankey and Sunburst diagrams. For details on using these analytics, see Analytics.



10.6 Flow tapping

Sample the data of a deployed flow and check how each processing step changes the messages, to test your configuration.

Flow tapping allows you to sample the data flowing in a flow and see how it processes the logs. It's useful to test the flow when you're configuring or adjusting **Processing Steps**.

> **Note**
>
> **Neither AxoRouter nor AxoConsole records any log data. The Flow tapping and Log tapping features only send a sample of the data to your browser using encrypted connections while the tapping is active. Only authorized users can use tapping.**

Prerequisites

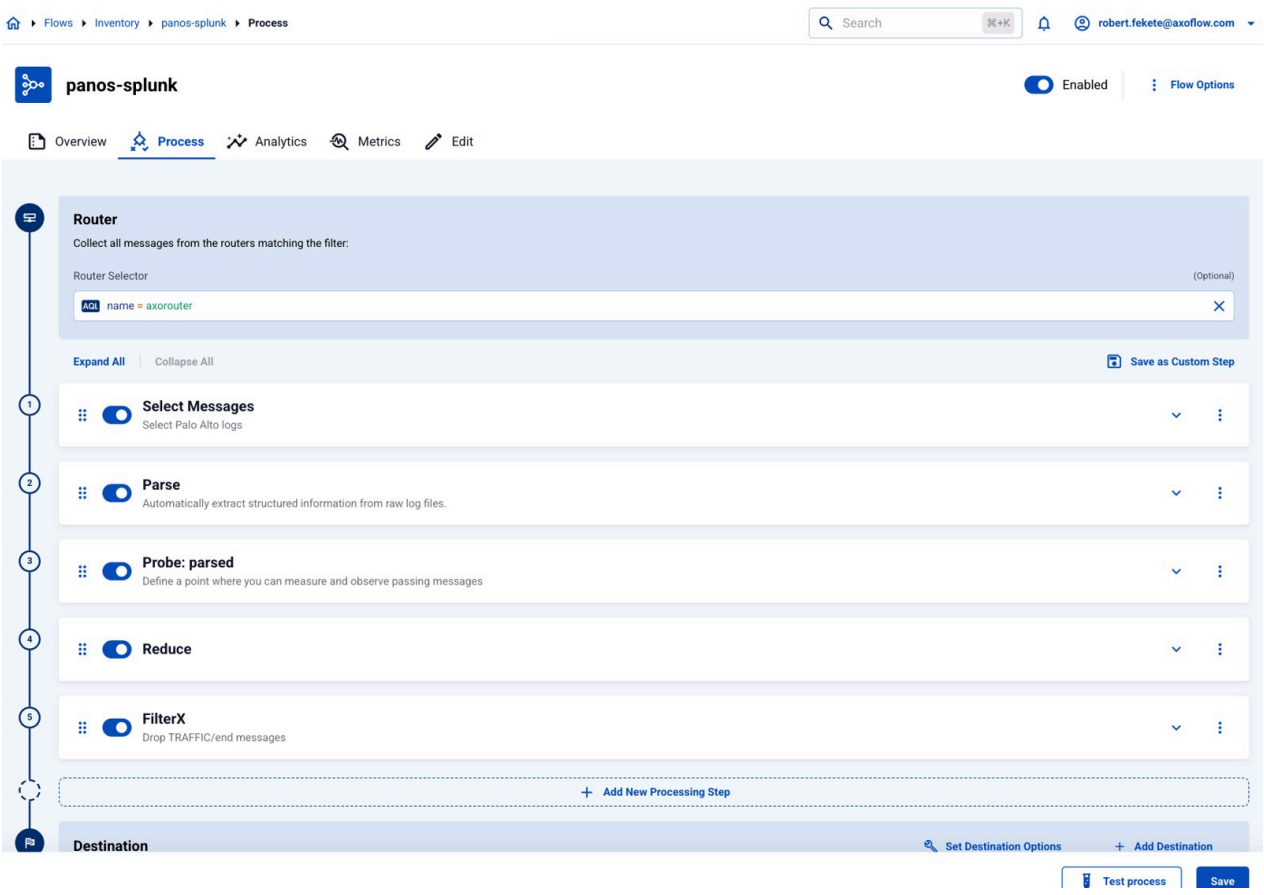
Flow tapping works only on flows that have been deployed to an AxoRouter instance. This means that if you change a processing step in a flow, you must first **Save** the changes. If the test panel is open, you can also select **Save & Update Test**.

If you want to experiment with a flow, we recommend that you clone the flow (select **more_vert > Clone** on the card of the flow) and configure it to use the /dev/null destination, then redo adjusting the final processing steps in the original flow.

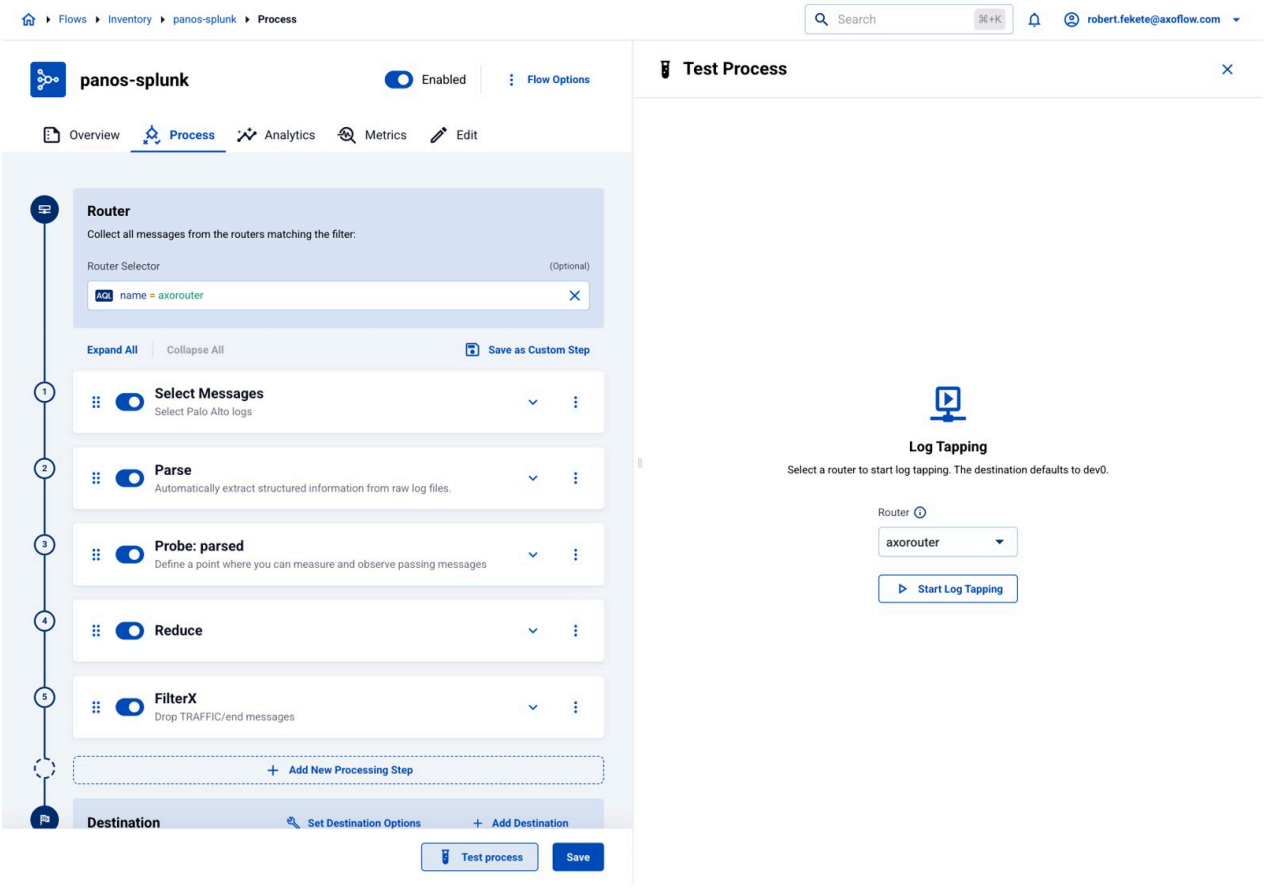
Tap into a flow

To tap into a Flow, complete the following steps.

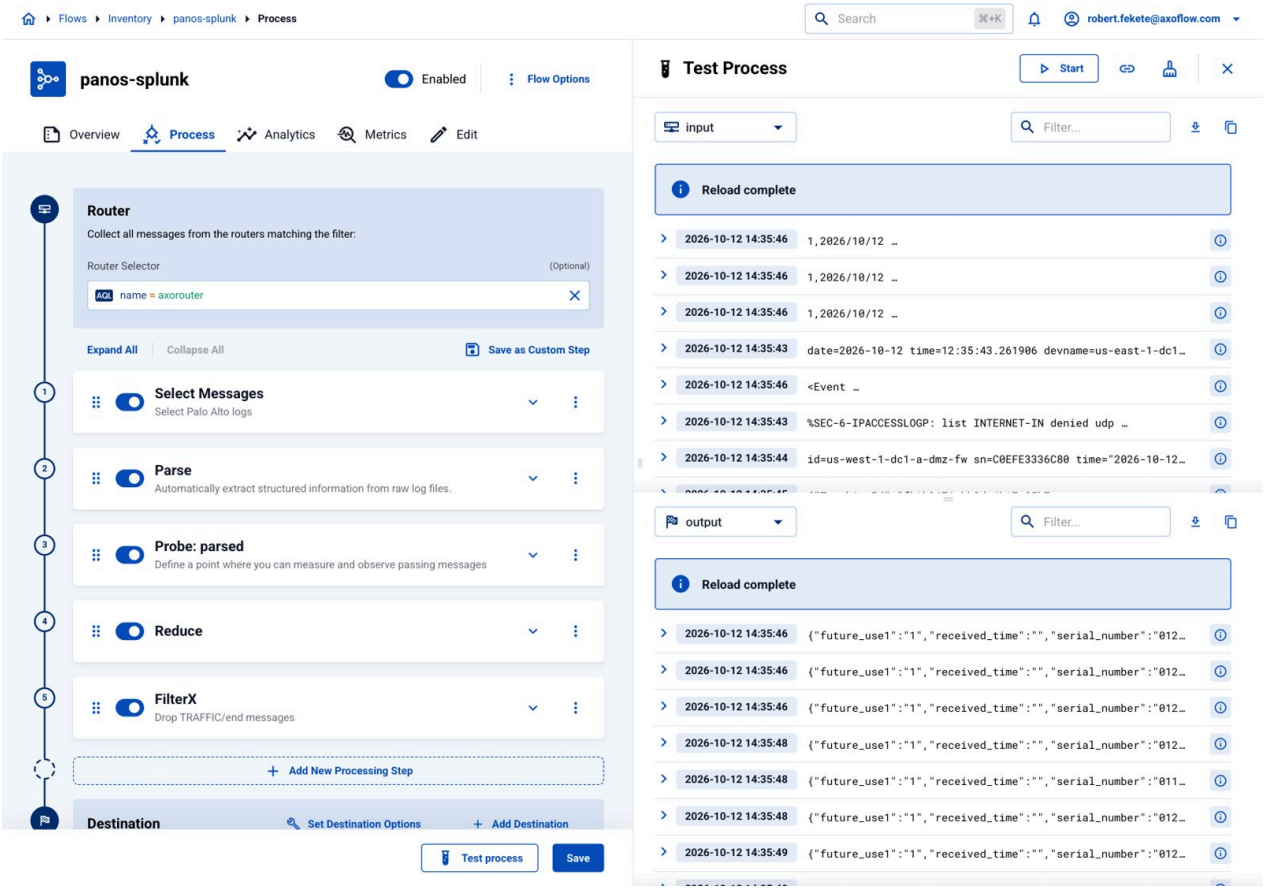
1. Open AxoConsole and select **Flows**.
2. Select the flow you want to tap into, then select **Process > Test process**.



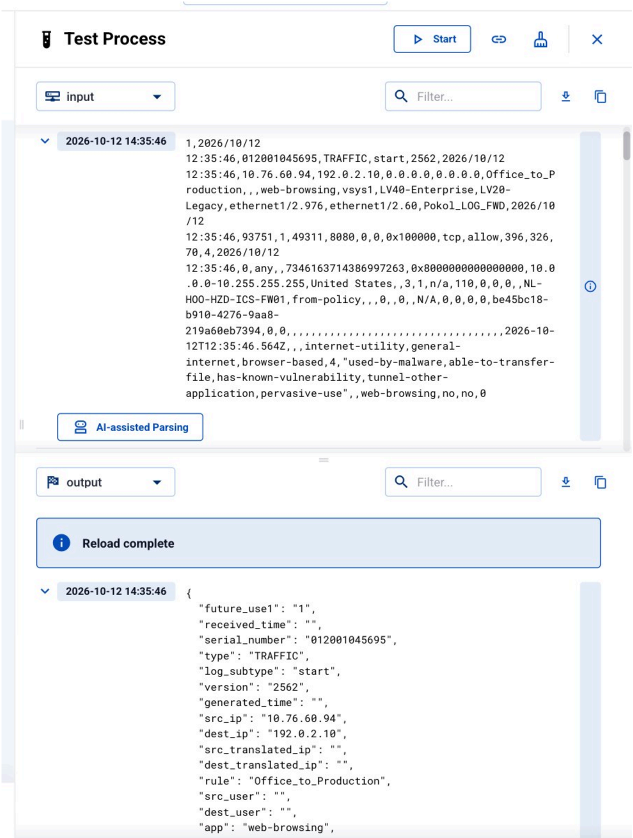
3. Select the router where you want to tap into the flow, then click **Start Log Tapping**.



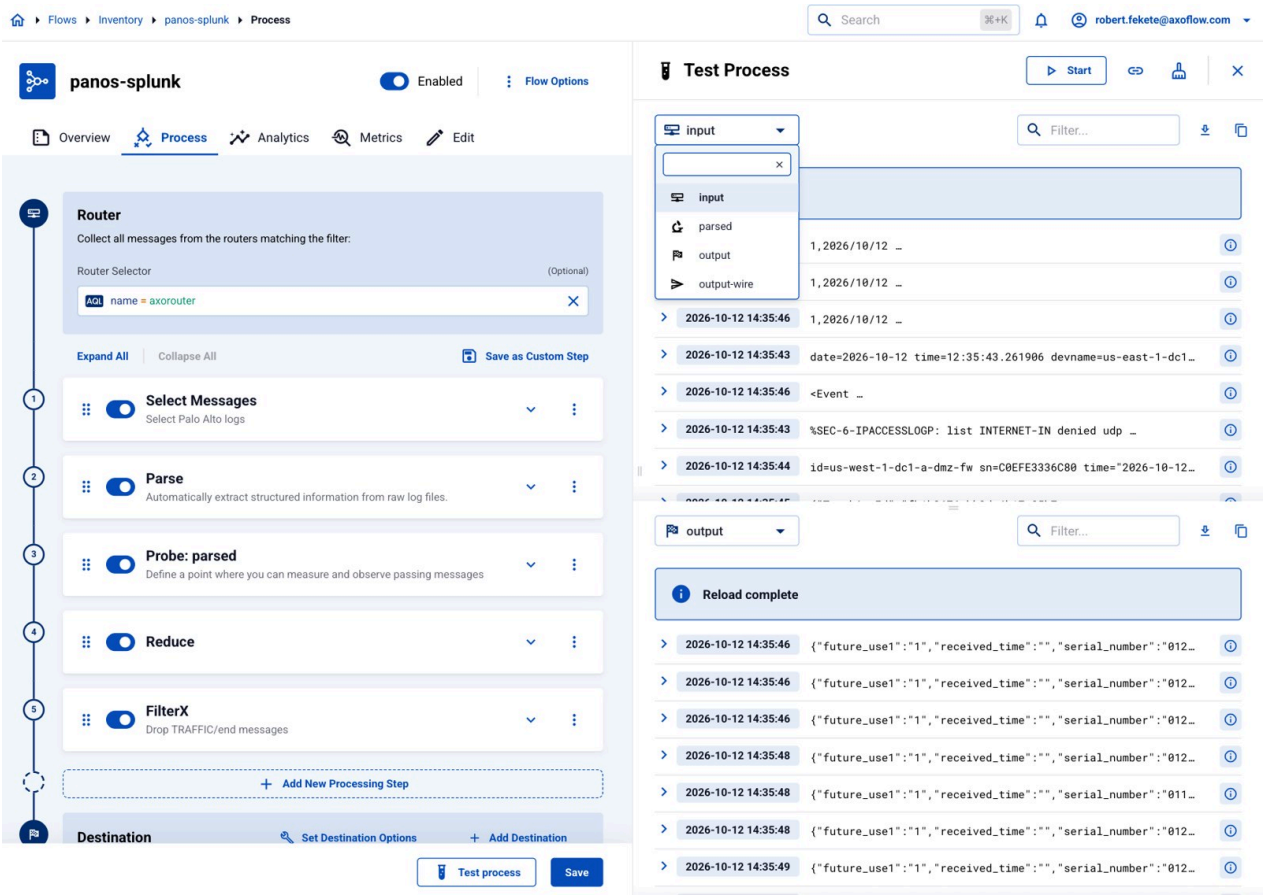
4. AxoConsole displays a sample of the logs from the data flow in matched pairs for the input and the output of the flow.



- The **input** point is after the source connector finished processing the incoming message (including parsing and classification performed by the connector).
- The **output** shows the messages right before they are formatted according to the specific destination's requirements. By default, the messages are linked: opening the details of a message in the output automatically opens the details of the message input window, and vice versa (provided the message you clicked exists in both windows). If you want to compare a message with other messages, click link to disable linking.
- Select **output-wire** to see the messages exactly as they're sent to the destination. See the **wire** fields in the details of the event. The **output-wire** point requires AxoRouter version 0.90.0 or newer.



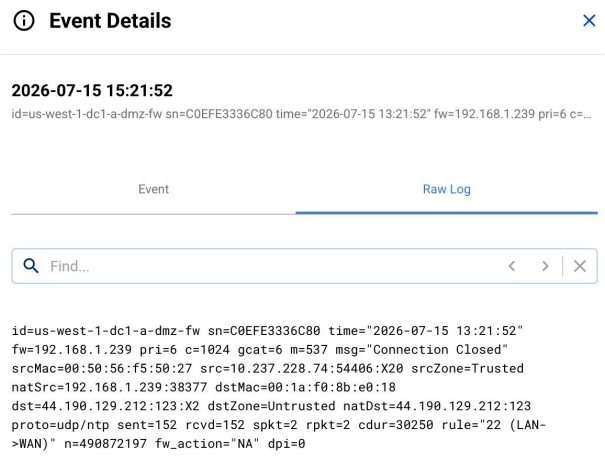
5. If the flow contains one or more **Probe** steps, you can select the probe as the tapping point. That way you can check how the different processing steps modify your messages. To change the tapping point of a pane, select it in the dropdown above the pane: **input**, **output**, **output-wire**, or the name of a probe.



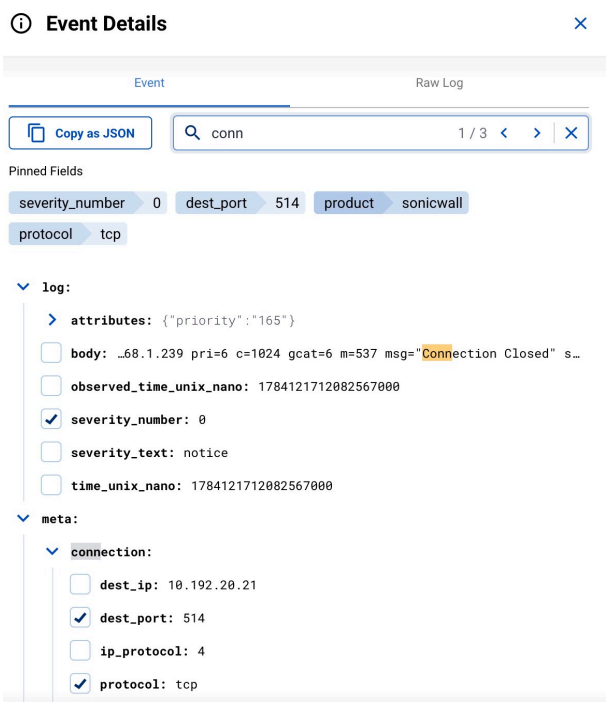
6. When the logs you're interested in show up, select **Pause**, then click the info icon next to a log message to see its details.

   The **Event Details** panel allows you to inspect a specific event. For details on the available fields and their structure, see Message schema reference.

   - To display the original log message, select **Raw Log**.

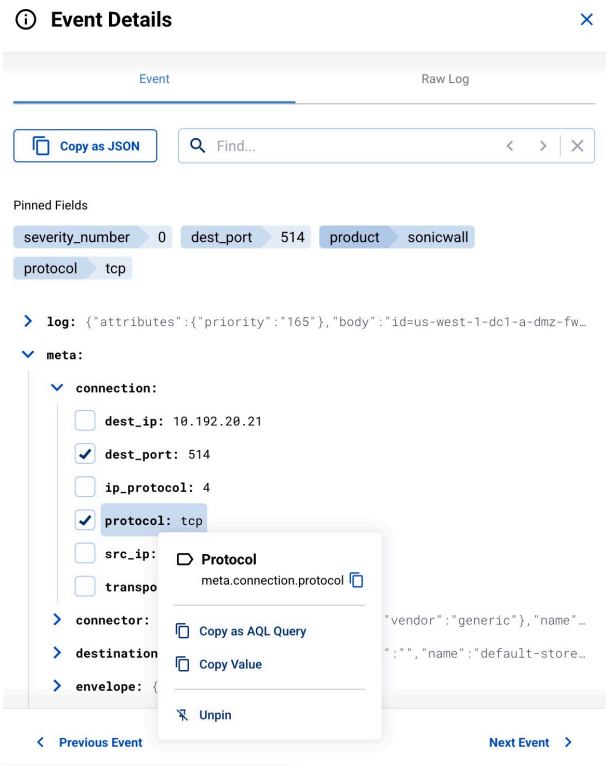


   - To copy the entire event as JSON to the clipboard, select **Copy as JSON**.
   - To search the event body and metadata, enter a search keyword into the **Find** field. This searches for the string on the **Event Details** panel, including the log, metadata, or label names and values.



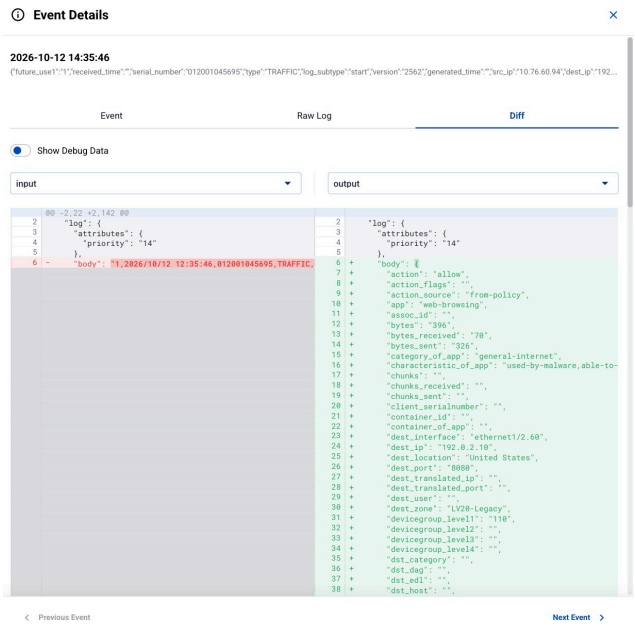
- To add a field to the **Pinned Fields** and display it for every event, click the checkbox before the field. Pinned fields show up in the event details, and also in the list of the events, making it easy to get a quick glance on the distribution of the field values. They're included as separate fields when you **Export** the search results into a CSV file.

  To unpin a field, select the field in any of the events, then select **Unpin** from the local menu.



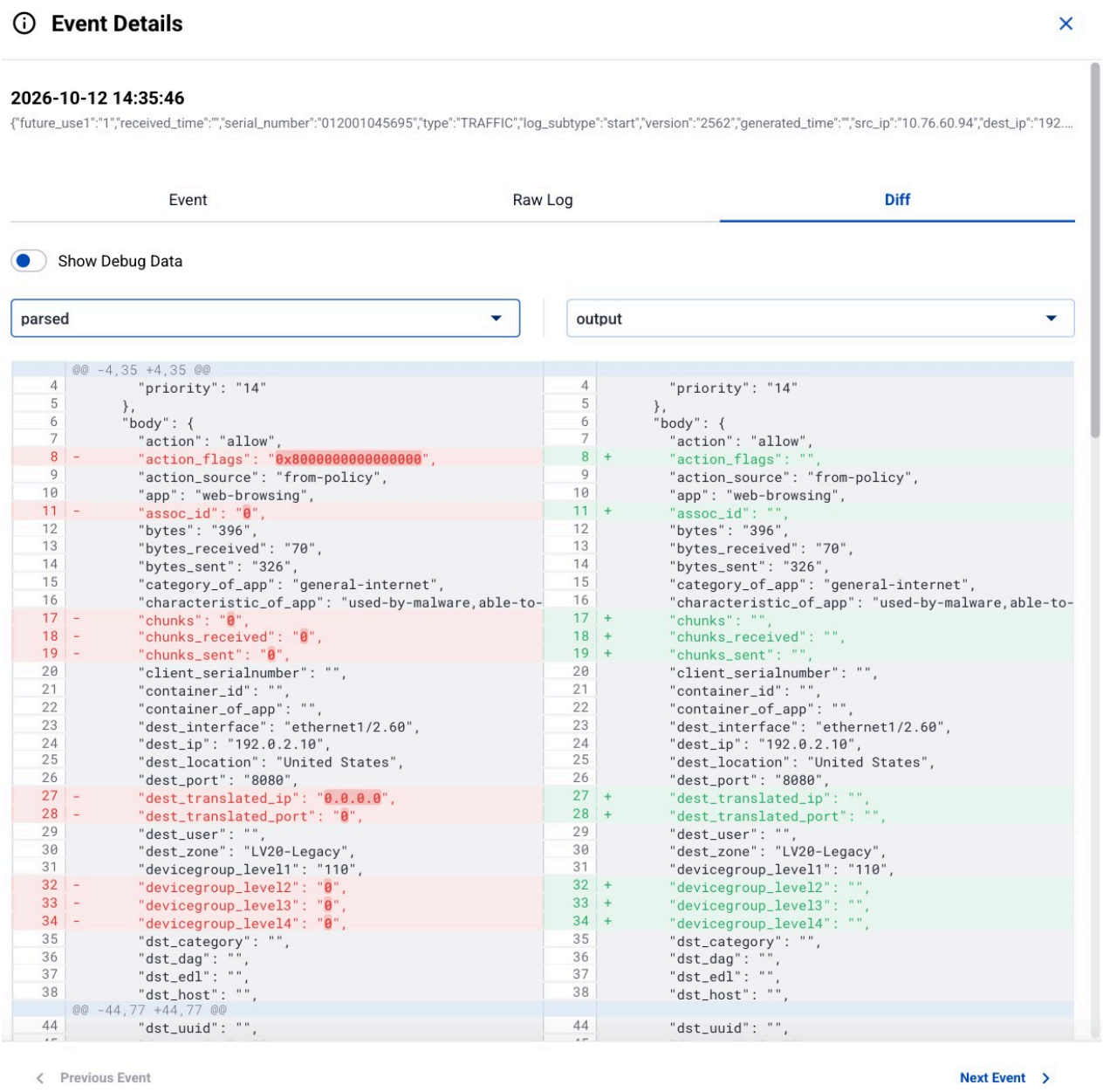
- To copy the name or the value of a field, select the name of the field, then select content_copy or **Copy Value**.
- To copy an AQL query that matches this field and its current value, select the name of the field, then select **Copy as AQL Query**. You can paste this query into the **AQL** search field, and use it for filtering.
- If the field refers to a pipeline element in Axoflow (for example, a flow in the `meta.flow` field, a store, or an AxoRouter), you can jump to the page of the related element.

7. Diff view. To display the line-by-line difference between the state of a message at two stages of processing, open the event details and select **Diff**.



The diff compares the internal JSON representation of the message and its metadata at two tapping points. By default, it compares the **input** and the **output** of the flow. In the previous screenshot, the **Parse** step replaced the initial string representation of the message body with a structured JSON object.

To compare other tapping points, select them in the two dropdowns: **input**, **output**, **output-wire**, or the name of a probe. You can select any pair. For example, the following screenshot compares the **parsed** probe with the **output**, so it shows only the changes of the steps after the probe.



Note

To parse a log message that AxoRouter can't parse automatically, expand the message in the input pane, then select AI-assisted Parsing. The AI assistant creates a FilterX processing step that parses the message. For details, see AI-assisted parsing.

Service status messages

While a tapping session is active, AxoConsole shows status messages about the tapped service inline in the log stream. They appear in every pane, regardless of the tapping point of the pane. These messages don't come from your data sources. Axoflow inserts them to tell you when the service is restarted or its configuration is reloaded, for

example, after you save a change to a flow or a connector rule. That way you can tell whether a change in the messages you see is the result of your configuration change, and notice when applying the new configuration failed.

The messages are color-coded by severity: informational messages (`notice` severity) are blue, errors (`err` severity) are red.

The following status messages can appear:

Message	Severity	Meaning
`Restarting service`	`notice`	AxoConsole has started restarting the service.
`Force restarting service`	`notice`	AxoConsole has started a forced restart of the service (the **Force restart** option of the **Restart** dialog on the **Services** tab of the host). A forced restart terminates the service without waiting for the delivery of in-flight messages.
`Service restart initiated successfully`	`notice`	The service has accepted the restart request.
`Restart failed: <details>`	`err`	The service couldn't be restarted.
`Reload complete`	`notice`	The service is running with the new configuration.
`Reload failed: <details>`	`err`	The service couldn't apply the new configuration, and keeps running with the previous one. Usually this means that the new configuration is invalid, for example, a FilterX processing step contains a syntax error.
`Reload blocked due to missing config files: <details>`	`err`	The service couldn't apply the new configuration because a file it references (such as a certificate) isn't available on the host.

Show debug data

The details of a tapped message can include a `debug` field that shows how AxoRouter represents the message internally:

- `debug.filterx` contains the FilterX variables that are set when the message reaches the tapping point.
- `debug.msg` contains the name-value pairs of the message.

These fields are hidden by default. To display them, open the details of a message, then select **Show Debug Data**. The switch is available only for messages that have debug data.

Debug data is useful when you're troubleshooting a processing step that doesn't behave as you expect, or when Axoflow support asks you for it. For the fields of the message itself, see Message schema reference.

10.7 Rehydrate data

Send selected data from AxoStore or AxoLake to a SIEM or another destination, for example, during an investigation.

Rehydrating data allows you to select a set of data stored in AxoStore and send it to any other destination. For example, that way you can send data stored in AxoLake to a SIEM for further analysis.

Start a new rehydration

To start a new rehydration job, complete the following steps.

1. Select **Rehydration > Start New Rehydration**.

   Alternatively, run a search on the **Search Logs** page, then select **Rehydrate**. AxoConsole copies the query, the **Store**, the **Router**, and the time range of the search into the new rehydration job.

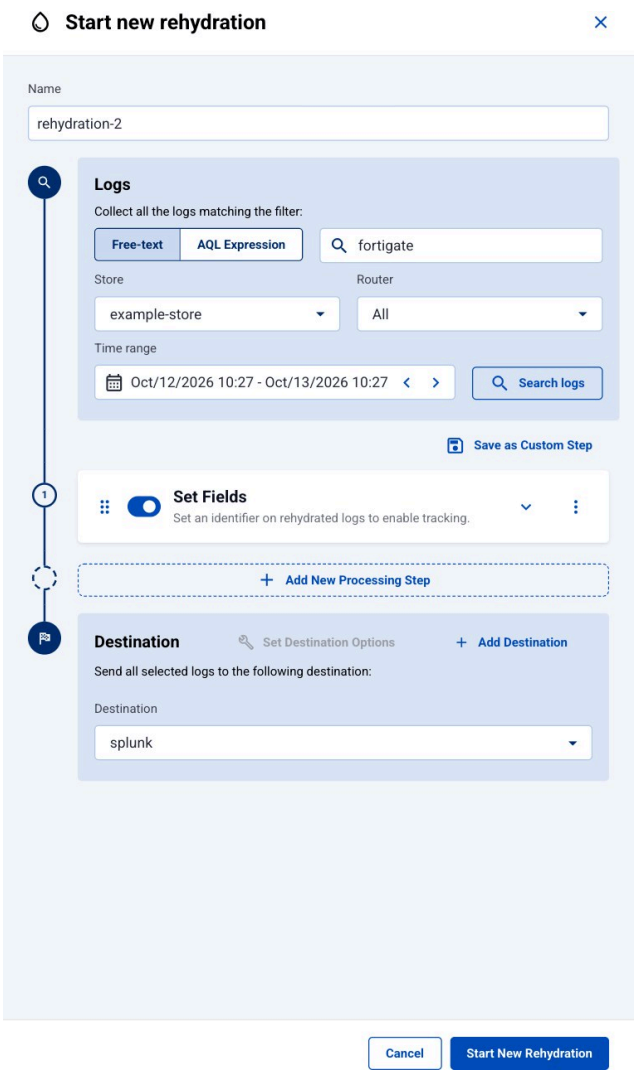


2. (Optional) AxoConsole automatically sets a name for the job. Set a custom name for the job if needed. This name is also added to the metadata of the rehydrated logs in the `meta.rehydration_name` field in a pre-filled processing step, which can be removed or modified if desired.

3. Specify which logs you want to rehydrate. Like on the **Search Logs** page, you can:

   - use **Free-text** or **AQL Expression** search
   - select logs from a specific **Store** or a specific **Router**
   - select logs from a specific time range.

   To check which logs match, select **Search logs**.

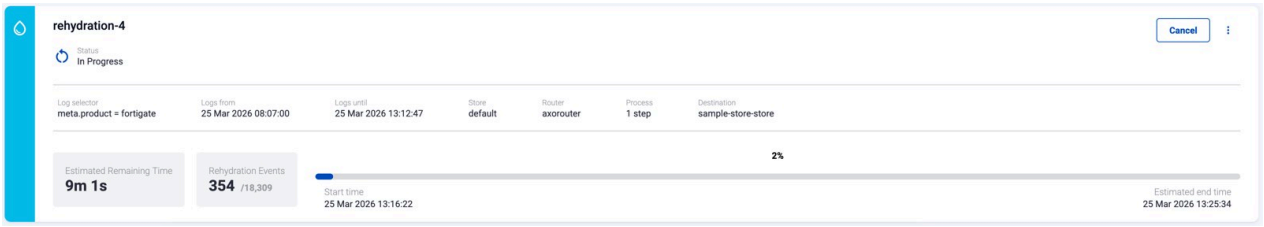


4. (Optional) Add processing steps if needed. To reuse the processing steps later, select **Save as Custom Step**.

5. Select the **Destination** where you want to send the rehydrated data.

6. Select **Start New Rehydration** to start the job.

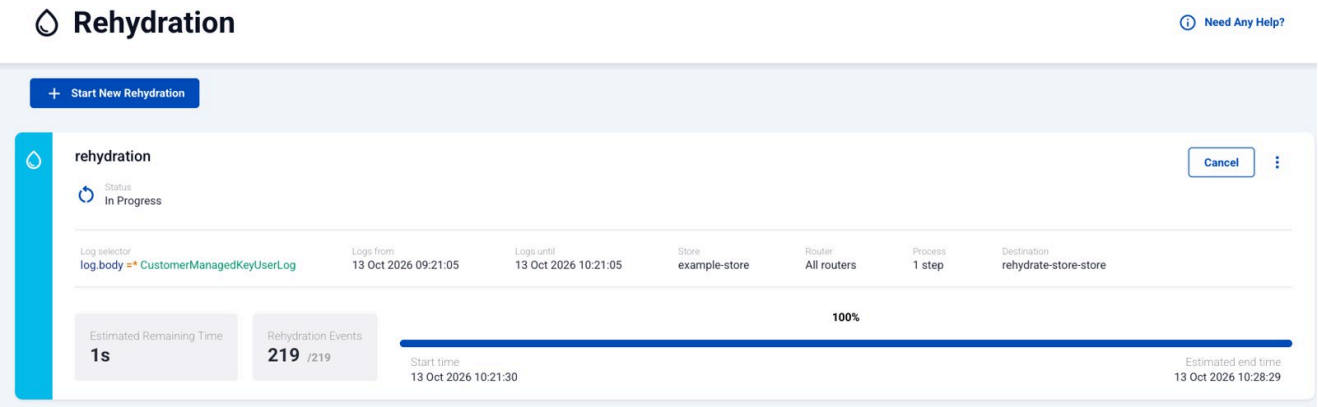
   AxoConsole configures and starts the job. You can follow its progress on the **Rehydration** page.



For details on the possible performance impact of rehydration, see Performance considerations.

Manage rehydration jobs

All existing rehydration jobs are listed on the **Rehydration** page.



For every job the following information is shown:

- **Name**: The name of the job.
- **Status**: `In Progress`, `Completed`, `Canceled` (stopped by a user), or `Failed`. For failed jobs, select **See failure details.** on the card to display the error.
- The data selectors used for the job: **Log selector**, **Logs from**, **Logs until**, **Store**, and **Router**, as well as the number of processing steps (**Process**) and the **Destination**.
- **Estimated Remaining Time** (only for running jobs).
- **Rehydration Events**: The number of events sent to the destination, and the number of events that matched the selectors.
- The progress bar of the job, with the **Start time** and the end time of the job. For running jobs, the end time is the **Estimated end time**. For canceled and failed jobs, it's the **Cancellation time** or the **Fail time**.

You can perform the following actions on the jobs:

- To cancel a running job, select **Cancel** on its card, then select **Terminate**. You can't continue a canceled job, but you can clone it.
- To clone a job (and optionally modify its parameters), select more_vert > **Clone**. For canceled jobs, you can also select **Clone and Restart** on the card.
- To delete a job, select more_vert > **Delete**. You can't delete a running job.

Performance considerations

Rehydration jobs use the same AxoRouter destinations that other flows might use. For example, if you're sending data to Splunk in a flow, and you're also rehydrating data to the same Splunk destination, then these operations use the same resources (like buffers and bandwidth). As a result, running rehydration on an AxoRouter under heavy load may cause backpressure towards the sources, and slow down data ingestion.

If the AxoRouter host has enough free resources, you can create a duplicate of the destination with a lower number of workers, and use that in the rehydration job. This can help to prevent rehydration jobs causing issues with normal workloads.

10.8 AI assistant

Use the AI assistant in AxoConsole to create FilterX parsers for log messages that AxoRouter doesn't parse by default.

The AI assistant in AxoConsole lets you complete data management and configuration tasks with a preconfigured large language model (LLM). The model, and the token and budget limits, depend on your subscription.

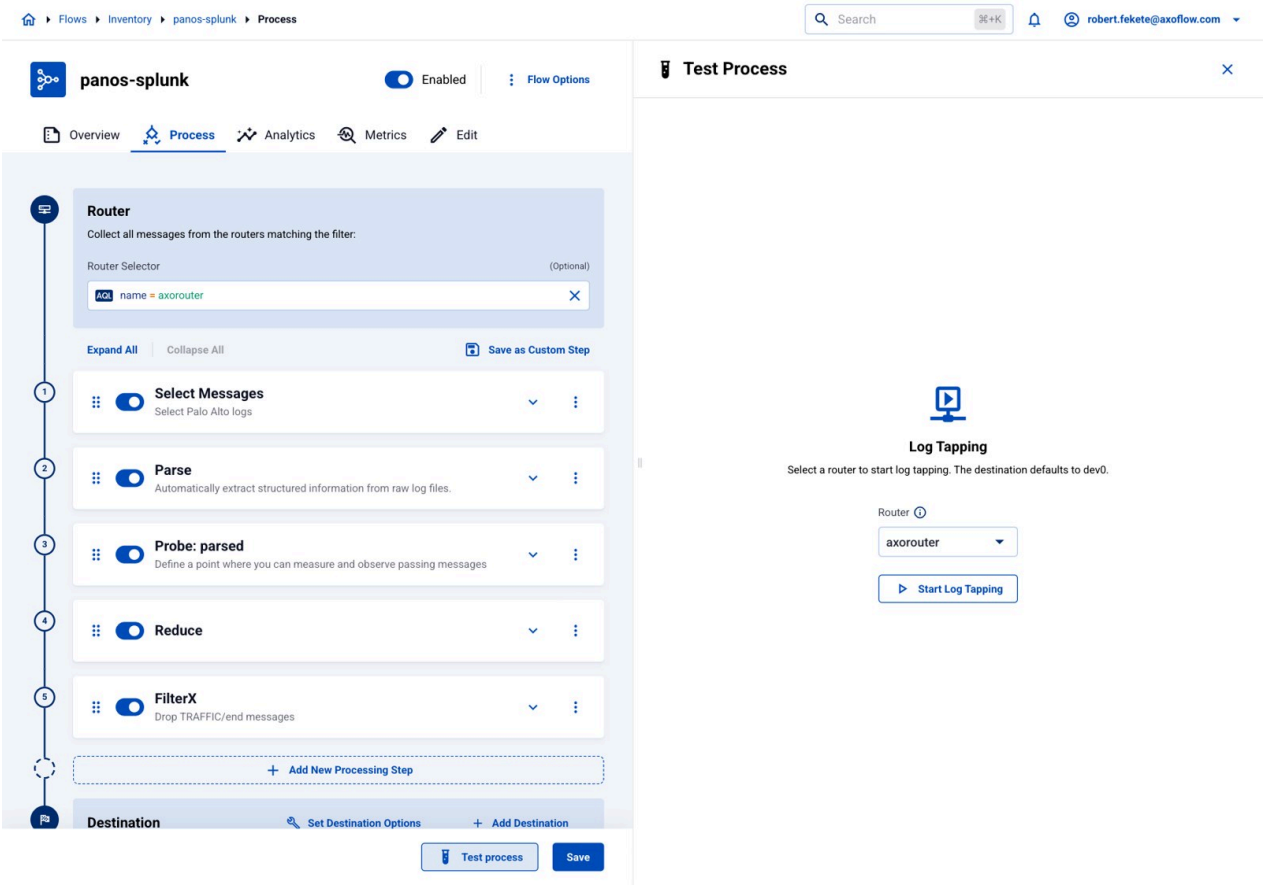
Prerequisites

- The AI assistant must be enabled and configured for your tenant. By default, it isn't enabled on every tenant. If you want to enable it for your tenant, contact our support team.
- For on-premises AxoConsole deployments, enabling the AI assistant requires custom integration. Contact our support team for details.

AI-assisted parsing

AI-assisted parsing creates a custom FilterX processing step to parse a specific log message. Use it for the logs of custom applications, or for log sources that AxoRouter doesn't support by default. Complete the following steps.

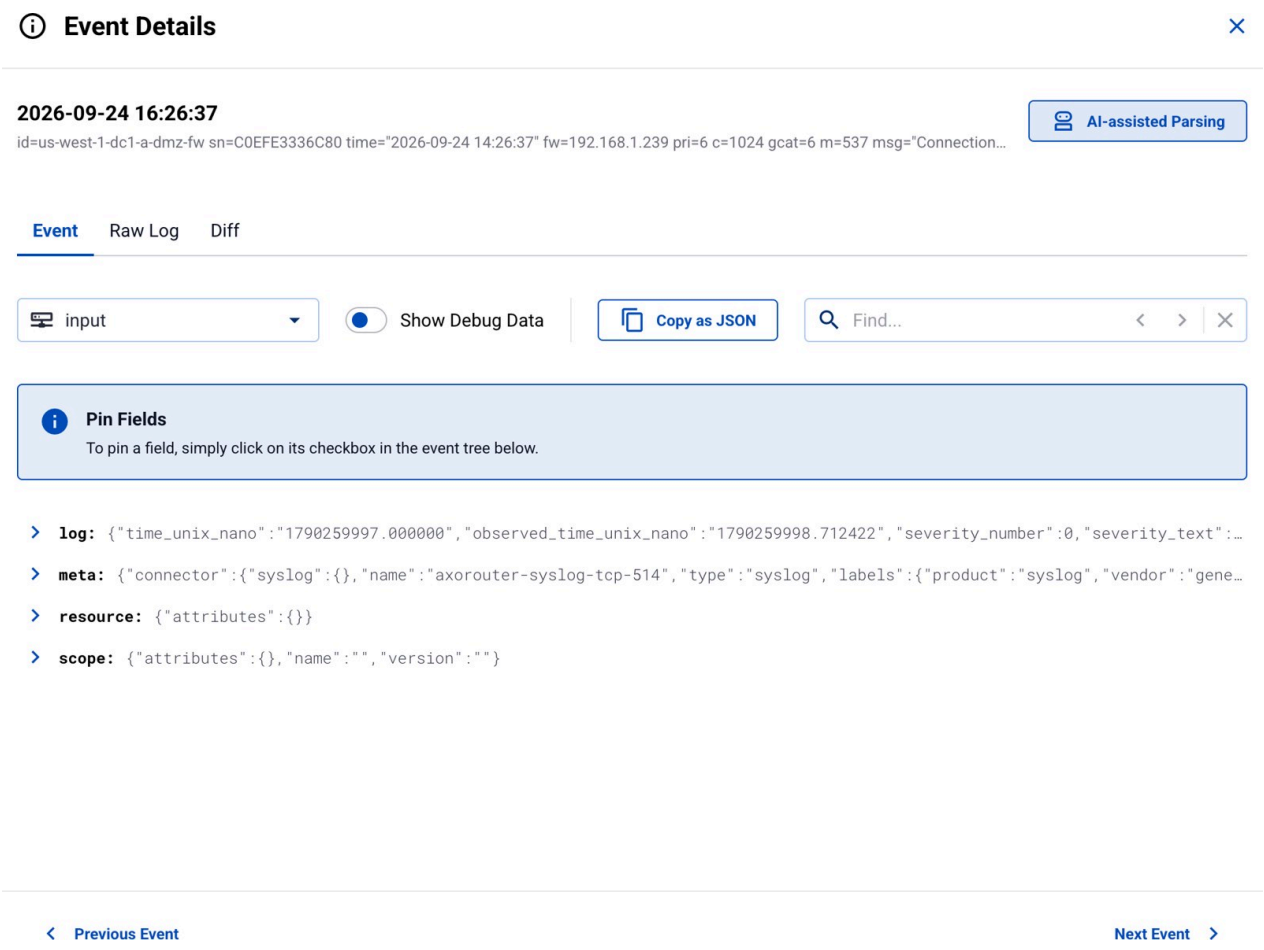
1. Use Flow tapping to find the log message you want to parse.
    1. Select **Flows**, then the flow you want to tap into
    2. Select **Process > Test process**.



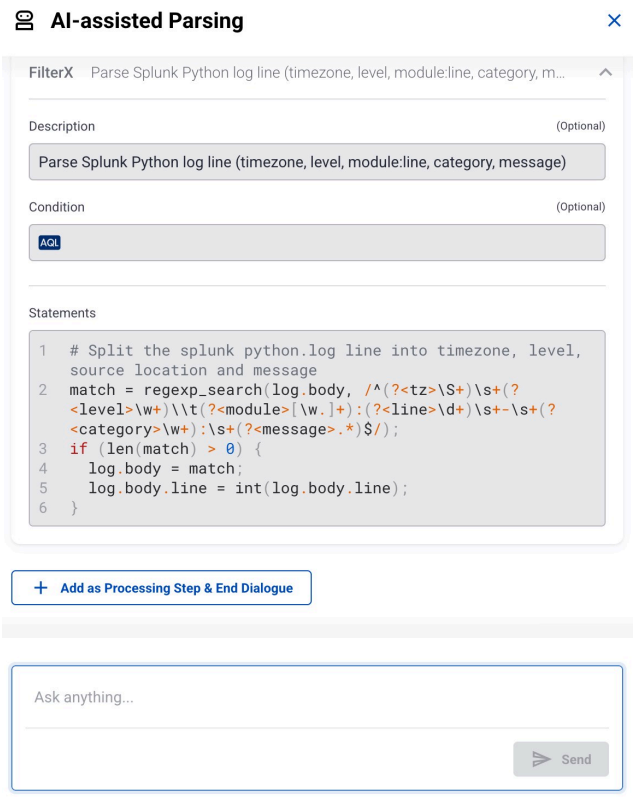
2. When the logs you're interested in appear, select **Pause**.

3. Select the info icon next to a log message to see its details.

4. Select **AI-assisted Parsing**. This option is available only for messages whose body (`log.body`) is an unparsed string.



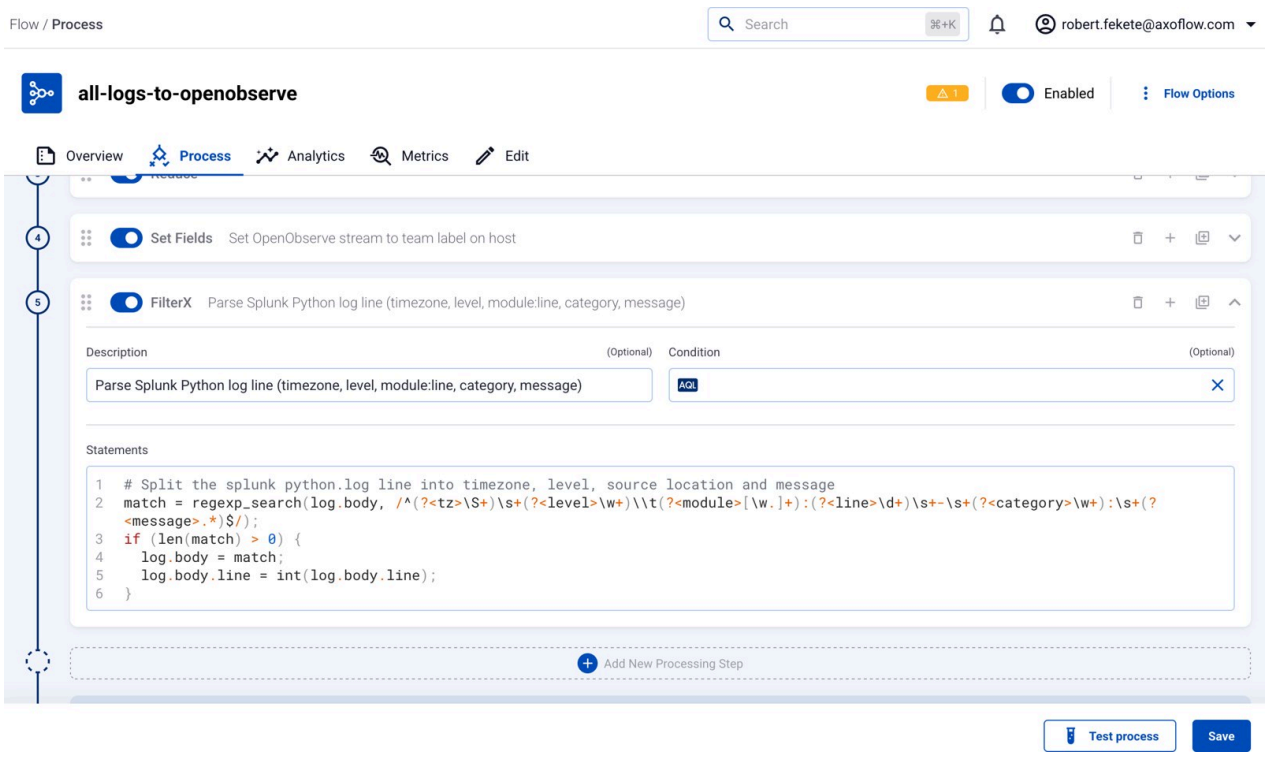
5. Wait until the AI assistant creates the parser.



6. If you want to adjust the parser, chat with the assistant.

7. If you're satisfied with the result, select **Add as Processing Step & End Dialogue**. The assistant adds the processing step to the flow.

8. Check that the processing step is in the right position, then select **Save** to save the flow.



11 Manage and monitor the pipeline

Manage and monitor the hosts of your pipeline: the host inventory, topology, dashboard, alerts, and log tapping.

When your pipeline is running, use these pages of AxoConsole to see what it does, and to find problems.

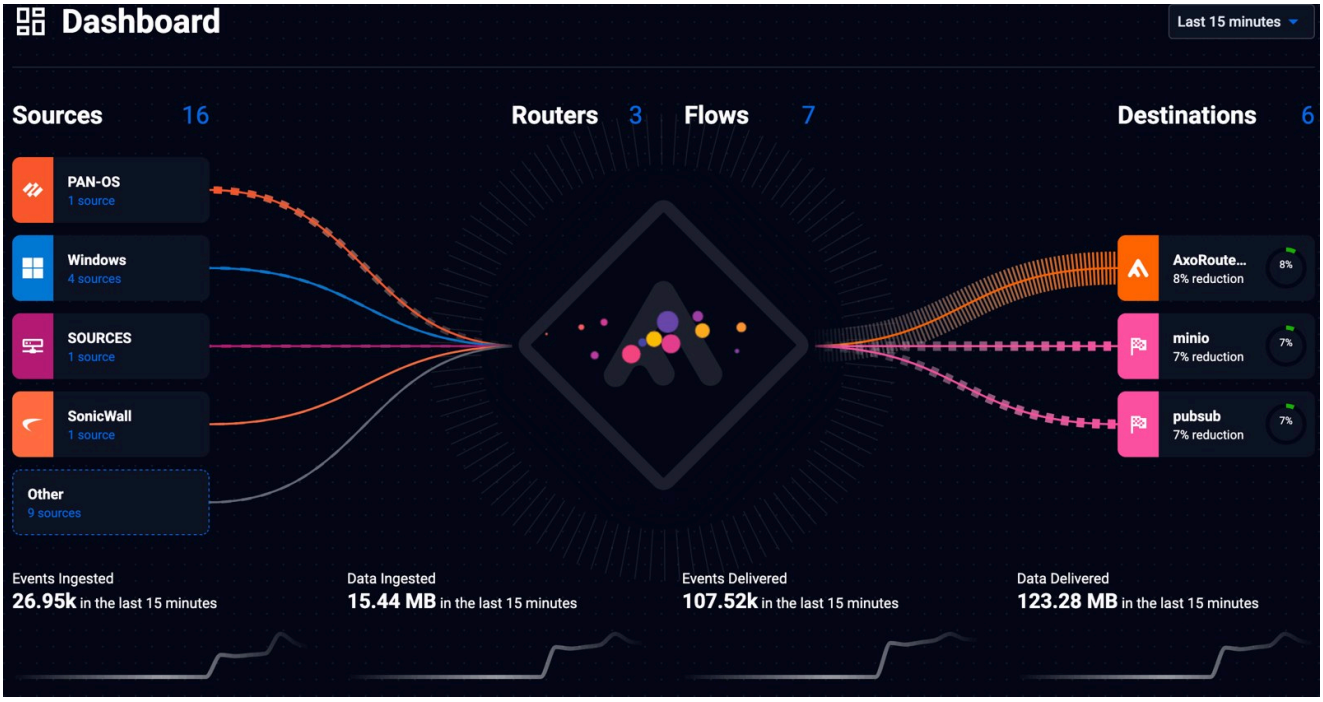
- The Dashboard summarizes your pipeline: the number of sources, routers, flows, and destinations, the data reduction, and the active alerts.
- The Topology page shows how data moves from your sources to your destinations. Select a host to see its details.
- The Hosts pages show your host inventory. You can add custom labels to hosts one by one, or in bulk.
- The Alerts page lists the problems that Axoflow detected in your pipeline.
- Log tapping shows you samples of the live data on a host.
- Private connections connect your cloud hosts to AxoConsole without using the public internet.

For metrics and analytics about your hosts and flows, see Metrics and analytics.

11.1 Dashboard

Read the AxoConsole dashboard: the counts of sources, routers, flows, and destinations, data reduction, ingested volume, and active alerts.

The **Dashboard** page gives you an overview of your pipeline, and links to relevant pages of pipeline elements. All data is automatically updated every 10 seconds.



The top row shows the number of **Sources**, **Routers**, **Flows**, and **Destinations**.

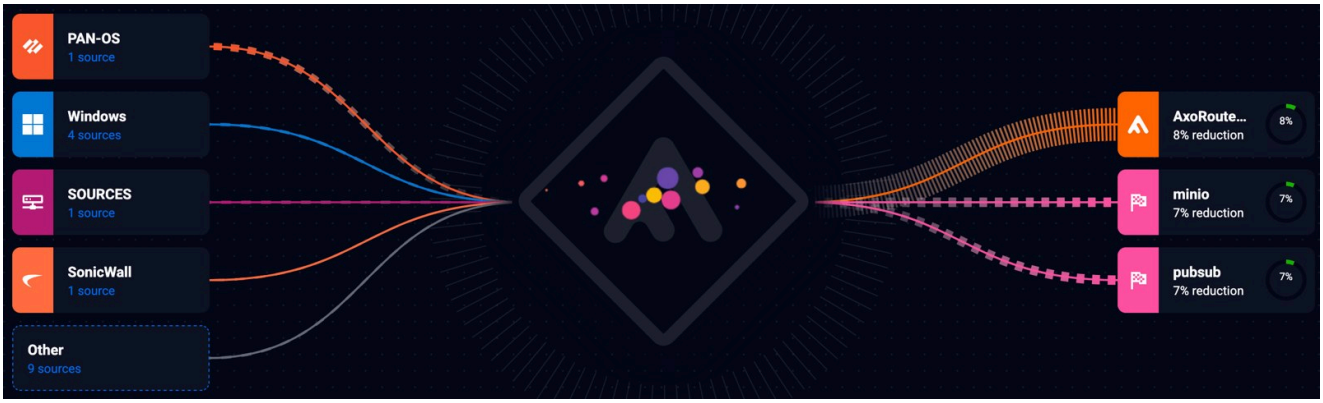


The animated section shows a simplified version of the **Topology** page.

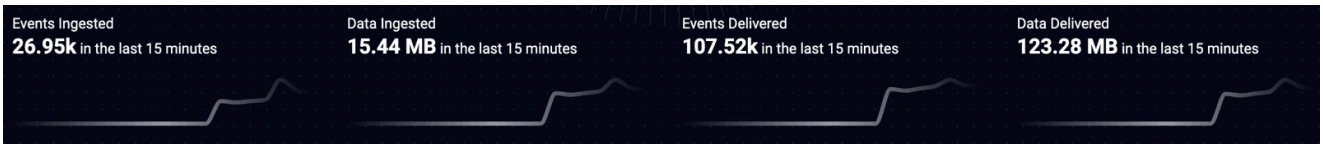
- The source side shows the most active source groups.
- The destination side shows the three most active destinations and the data reduction achieved.

  Data reduction is calculated as follows:

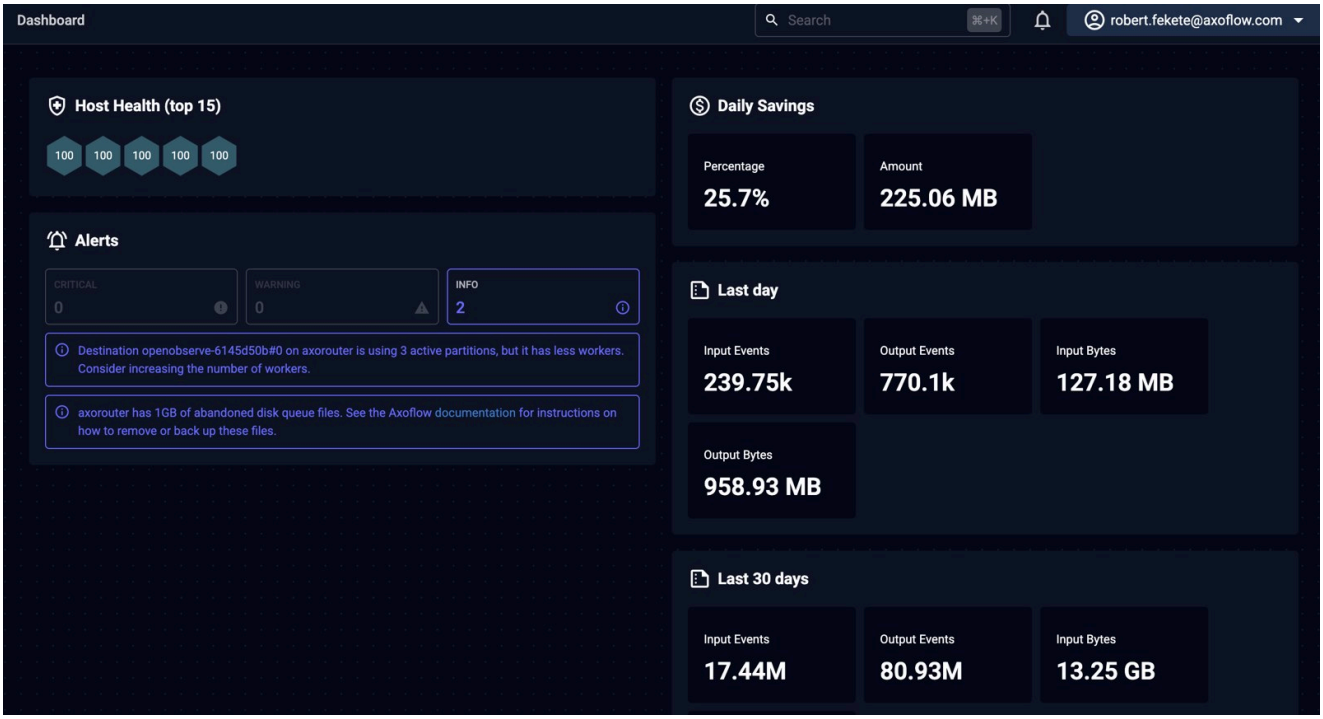
  - The input is the total data of every flow that sends data to the destination, measured either: before classification and parsing, or at the Probe processing step called `filtered` if it exists in the flow.
  - The reduced data is the total amount of data (in bytes) sent to the destination.
  - The data reduction is the ratio of the reduced data and the input.



The metrics bar shows the number of events and amount of data ingested from the sources and delivered to the destinations during the selected time frame. (You can change the time frame in the top-right of the page.) Note that the small charts under the metrics show the historic value of the related metric for a longer period, not the just within the selected time frame, so the current numbers correspond to the endpoint of the chart.



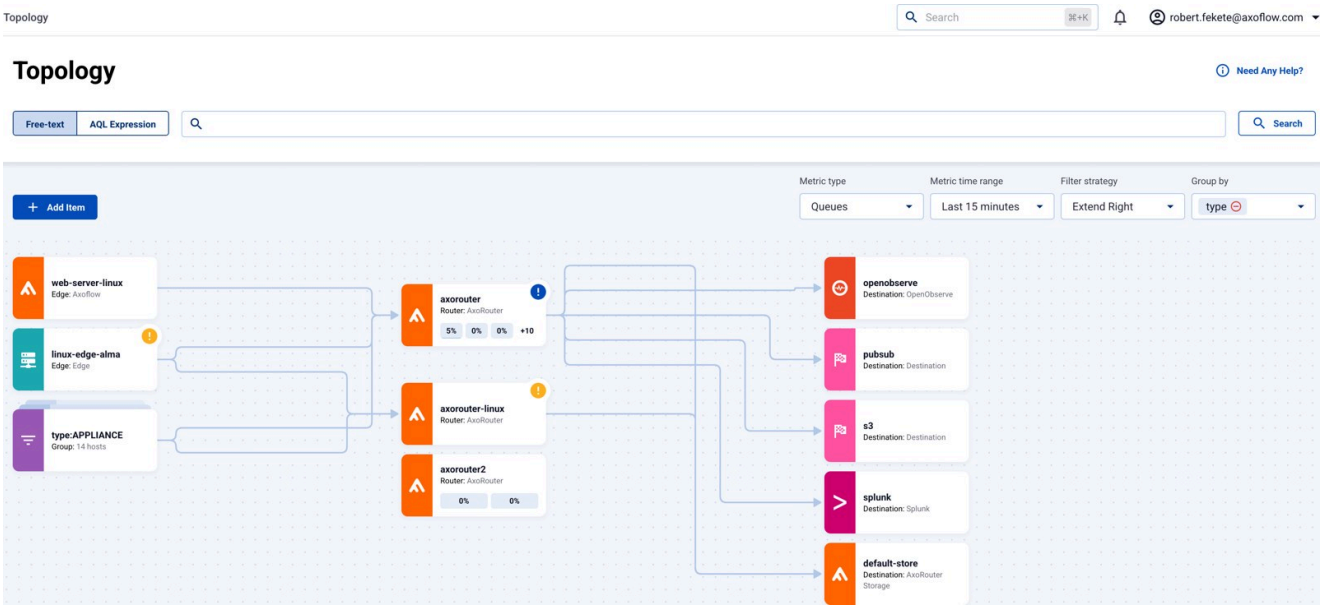
Further down, the dashboard shows the active alerts (if any), and metrics for the last day and the last 30 days.



11.2 Topology

See how data moves from your sources through AxoRouter to your destinations, and drill down to the hosts that have problems.

Based on the collected metrics, Axoflow visualizes the topology of your security data pipeline. The topology allows you to get a semi-real-time view of how your edge-to-edge data flows, and drill down to the details and metrics of your pipeline elements to quickly find data transport issues.



Select the name of a source host or a router to show the details and metrics of that pipeline elements.

If a host has active alerts, it's indicated on the topology as well.



Traffic volume

Select **Metric type > Log Bytes** or **Metric type > Log Events** in the top bar to show the volume of the data flow on the topology paths in bytes per second or events per second. The metrics refer to the last 15 minutes by default, select **Metric time range** to change to a different period.

Filter hosts

To find or display only specific hosts, you can use the filter bar.

- **Free-text** mode searches in the values of the following fields of the host: **Name**, **IP Address**, **GUID**, **FQDN**, and the labels of the host.

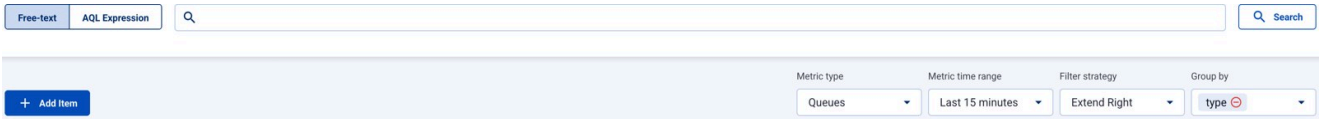
  The search is case insensitive. Adding multiple keywords searches for matches in any of the previous fields. This is equivalent to the `@ANY =* keyword1 AND @ANY =* keyword2` AQL query.

- **AQL Expression** mode allows you to search in specific fields or the labels of the hosts.

  It also makes more complex filtering possible, using the Equals, Contains (partial match), and Match (regular expression match) operators. Note that:

  - To execute the search, click **Search**, or hit ESC then ENTER.
  - AxoConsole autocompletes the built-in and custom labels and field names, as well as their most frequent values, but doesn't autocomplete labels and variables created by data parsing and processing steps.
  - You can use the AND and OR operators to combine expressions, and also parenthesis if needed. For details on AQL, see AQL operator reference.
  - The precedence of the operators is the following: parentheses, AND, OR, comparison operators.
  - Use the usual keyboard shortcuts to undo (⌘/Ctrl + Z) or redo (⌘/Ctrl + Shift + Z) your edits.

  For details about the message schema and the available fields, see Message schema reference.

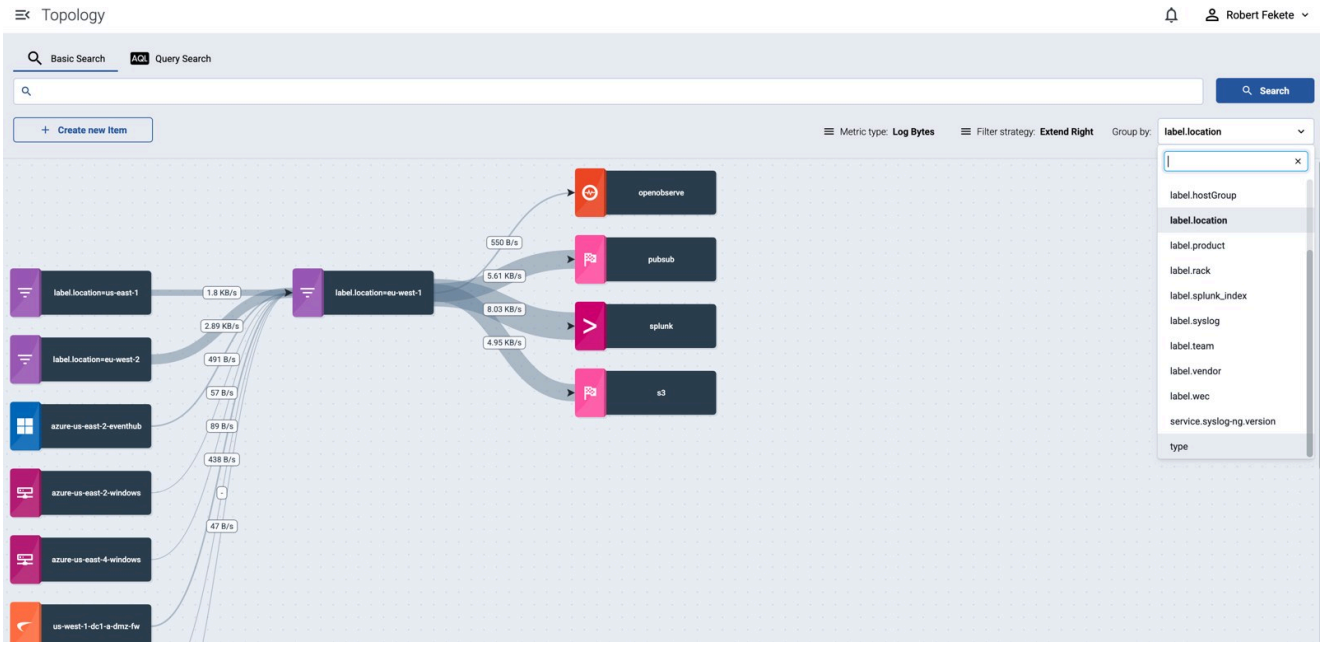


If a filter is active, by default the AxoConsole will display the matching hosts and the elements that are downstream from the matching hosts. For example, if an aggregator like AxoRouter matches the filter, only the aggregators and destinations where the matching host is sending data are displayed, the sources that are sending data to the matching aggregator aren't shown. To display only the matching host without the downstream pipeline elements, select **Filter strategy > Strict**. (For example, the AQL expression `type = ROUTER` shows only the aggregator hosts.)

The settings of the filter bar change the URL parameters of the page, so you can bookmark it, or share a specific view by sharing the URL.

Grouping hosts

You can select labels to group hosts in the **Group by** field, so the visualization of large topologies with lots of devices remains useful. AxoConsole automatically adds names to the groups. The following example shows hosts grouped by their region based on their location labels.

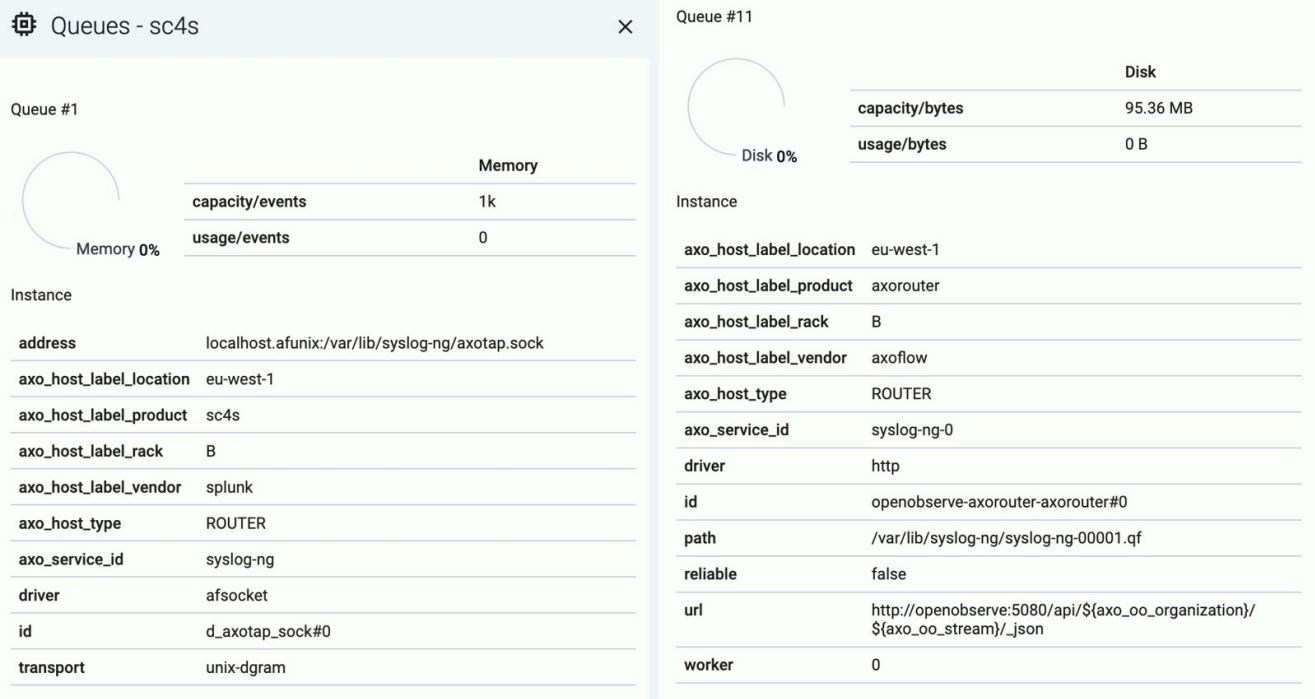


You can select one or more features to group hosts by. When multiple features are selected, groups will be formed by combining values of these features. This can be useful in cases where your information is split into multiple features, for example, if you have `region` (with values `eu`, `us`, etc.) and `zone` (with values `central-1`, `east-1`, `east-2`, `west-1`, etc.) labels on your hosts and select the `label.region` and `label.zone` features for grouping, you'll groups will be the following:

- A cross-product of the selected feature values: `eu,central-1`, `eu,east-1`, `eu,east-2`, `eu,west-1`, `us,central-1`, `us,east-1, and so on. Obviously, in this example, several of these groups (like` eu,east-1 `,` eu-east-2, and `us,central-1`) will be empty, unless your hosts are mislabeled.
- Groups for the individual values, for hosts that have only one of the selected features: `eu`, `us`, `central-1`, `east-1`, `east-2`, `west-1`, etc.

Queues

Select **Metric type > Queues** in the top bar to show the status of the memory and disk queues of the hosts. Select the status indicators of a host to display the details of the queues.



The following details about the queue help you diagnose which connections of the host are having throughput or connectivity issues:

- **capacity**: The total capacity of the buffer in bytes.
- **usage**: How much of the total capacity is currently in use.
- **driver**: The type of the AxoSyslog driver used for the connection the queue belongs to.
- **id**: The identifier of the connection.

Disk-based queues also have the following information:

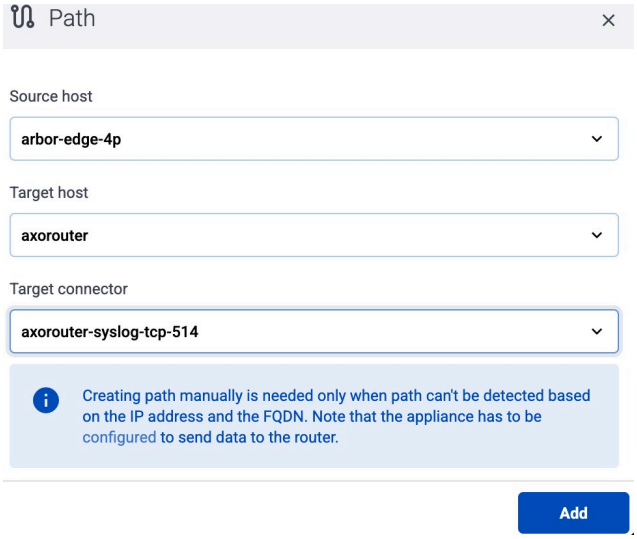
- **path**: The location of the disk-buffer file on the host.
- **reliable**: Indicates wether the disk-buffer is set to be reliable or not.
- **worker**: The ID of the AxoSyslog worker thread the queue belongs to.

Both memory-based and disk-based queues have additional details that depend on the destination.

Add a path

Paths show where the data flows on the **topology** page, for example, between a source and an AxoRouter. Usually, AxoConsole creates and deletes paths automatically. Sometimes you've to add a path manually, for example, when AxoConsole can't detect the path based on the IP address and the FQDN of the host. To add a path manually, complete the following steps.

1. Select **Topology > Add Item > Path**.



2. If not set automatically, select the beginning of the path in the **Source host** field.
3. Select the target router or aggregator this source is sending its data to in the **Target host** field, for example, `axorouter`.
4. Select the **Target connector**. The connector determines how the destination receives the data (for example, using which protocol or port).
5. Select **Add**. The new path appears on the **Topology** page.



11.3 Hosts

Find hosts, read their details and services, and add custom labels to sources, edge hosts, and AxoRouter instances.

Axoflow collects and shows you a wealth of information about the hosts of your security data pipeline. Sources and edge hosts are listed on the **Sources** page, while AxoRouters are shown on the **Routers** page.

- Sources are hosts that are sending data to a data aggregator, like AxoRouter.
- Edges are source hosts that are running a collector agent managed by AxoConsole, or have an Axolet agent reporting metrics from the host.

To tap into the logs of the host, see Log tapping.

11.3.1 Find a host

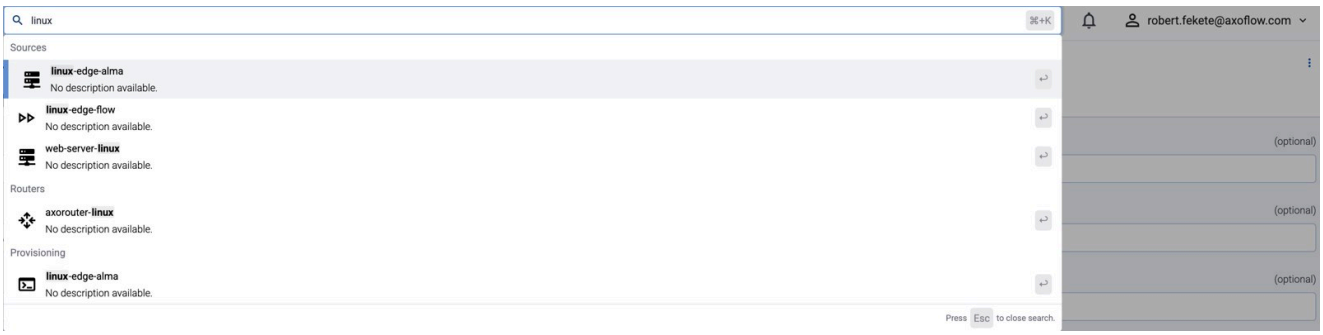
Find a host with global search, on the Topology page, or on the Routers and Sources pages.

To find a specific host, you can use any of the following methods:

- the global search,
- the **Topology** page, or
- the **Routers** or **Sources** pages.

Use global search

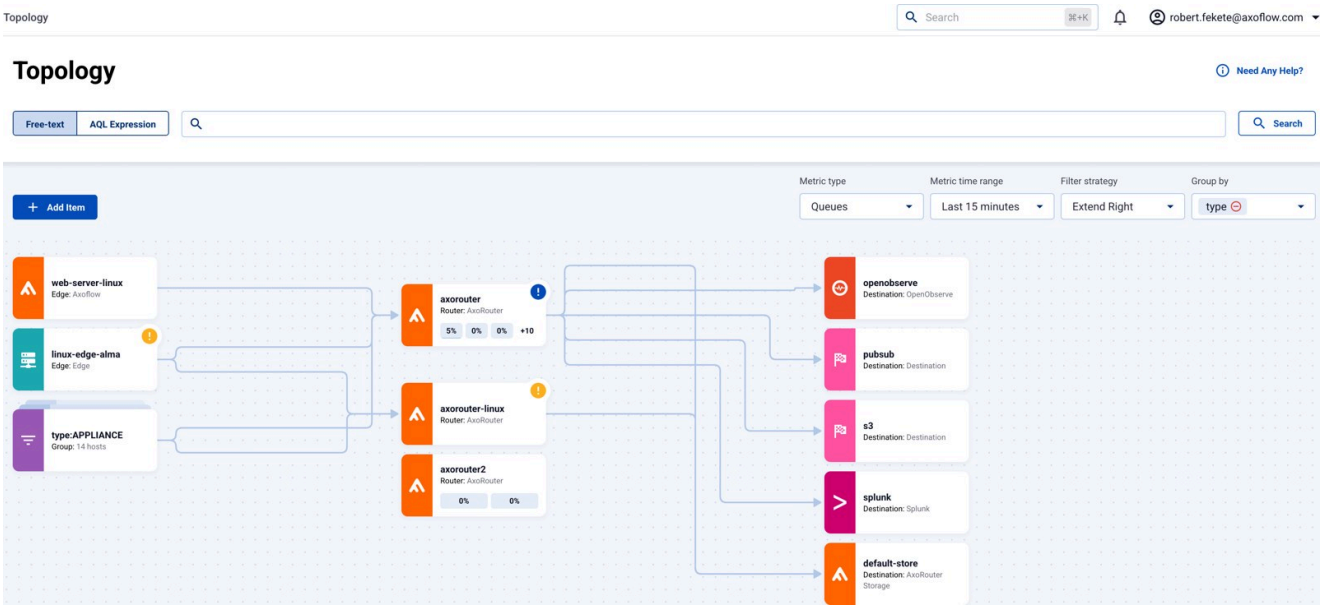
Select `⌘/Ctrl + K`, start typing the name of the host, then select the host from the results. For details, see Global search.



Find host on the Topology page

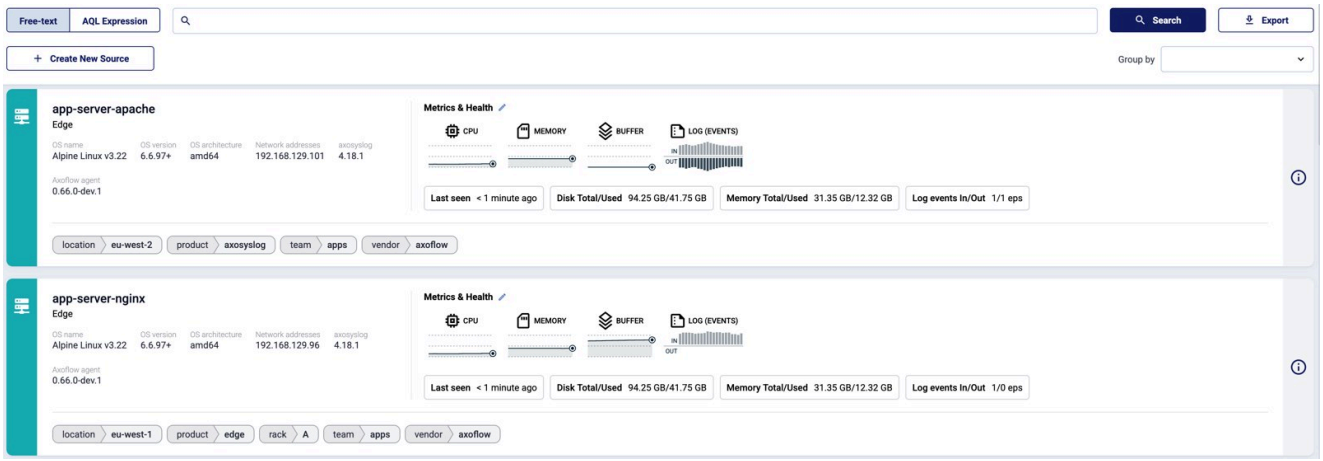
Open the **Topology** page, then click the name of the host you're interested in.

If you have many host and it's difficult to find the one you need, use filtering, or grouping.



Find a router or a source host

To find an AxoRouter, open the **Routers** page. To find a source or an edge host, open the **Sources** page.



To find or display only specific hosts, you can use the filter bar.

- **Free-text** mode searches in the values of the following fields of the host: **Name**, **IP Address**, **GUID**, **FQDN**, and the labels of the host.

  The search is case insensitive. Adding multiple keywords searches for matches in any of the previous fields. This is equivalent to the `@ANY =* keyword1 AND @ANY =* keyword2` AQL query.

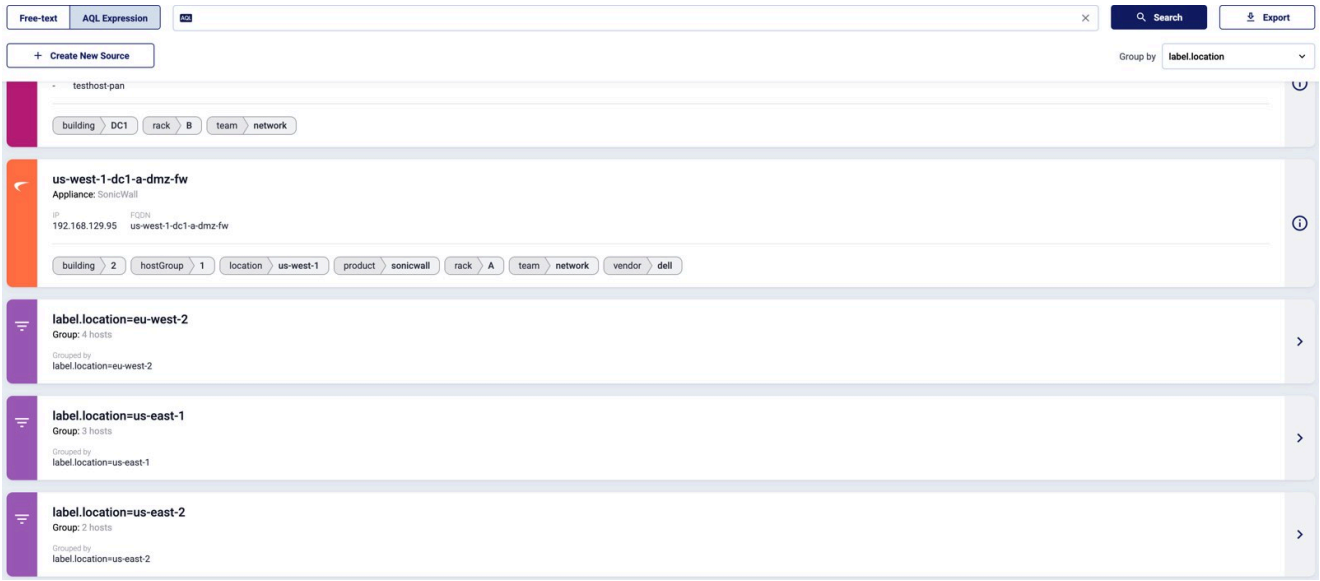
- **AQL Expression** mode allows you to search in specific fields or the labels of the hosts.

  It also makes more complex filtering possible, using the Equals, Contains (partial match), and Match (regular expression match) operators. Note that:

- To execute the search, click **Search**, or hit ESC then ENTER.
- AxoConsole autocompletes the built-in and custom labels and field names, as well as their most frequent values, but doesn't autocomplete labels and variables created by data parsing and processing steps.
- You can use the AND and OR operators to combine expressions, and also parenthesis if needed. For details on AQL, see AQL operator reference.
- The precedence of the operators is the following: parentheses, AND, OR, comparison operators.
- Use the usual keyboard shortcuts to undo (⌘/Ctrl + Z) or redo (⌘/Ctrl + Shift + Z) your edits.

For details about the message schema and the available fields, see Message schema reference.

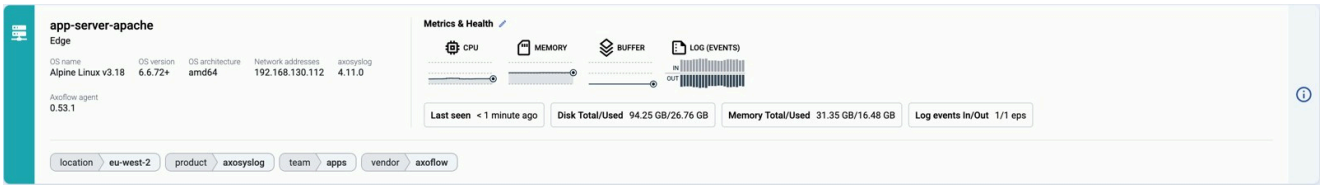
You can also select one or more features (for example, `label.location`) to group hosts by in the **Group by** field. For details on how grouping works, see Grouping hosts.



11.3.2 Host information

The hostname, labels, and other details that AxoConsole shows about sources, edge hosts, and AxoRouter instances.

AxoConsole provides a quick overview of every data source and edge host on the **Sources** page, and AxoRouter on the **Routers** page. The exact information depends on the type of the host: hosts managed by Axoflow provide more information than external hosts.



The following information is displayed:

- Hostname or IP address
- Metadata labels. These include labels added automatically during the Axoflow curation process (like product name and vendor labels), as well as any custom labels you've assigned to the host.

For edge hosts (hosts that have the Axoflow agent (Axolet) installed):

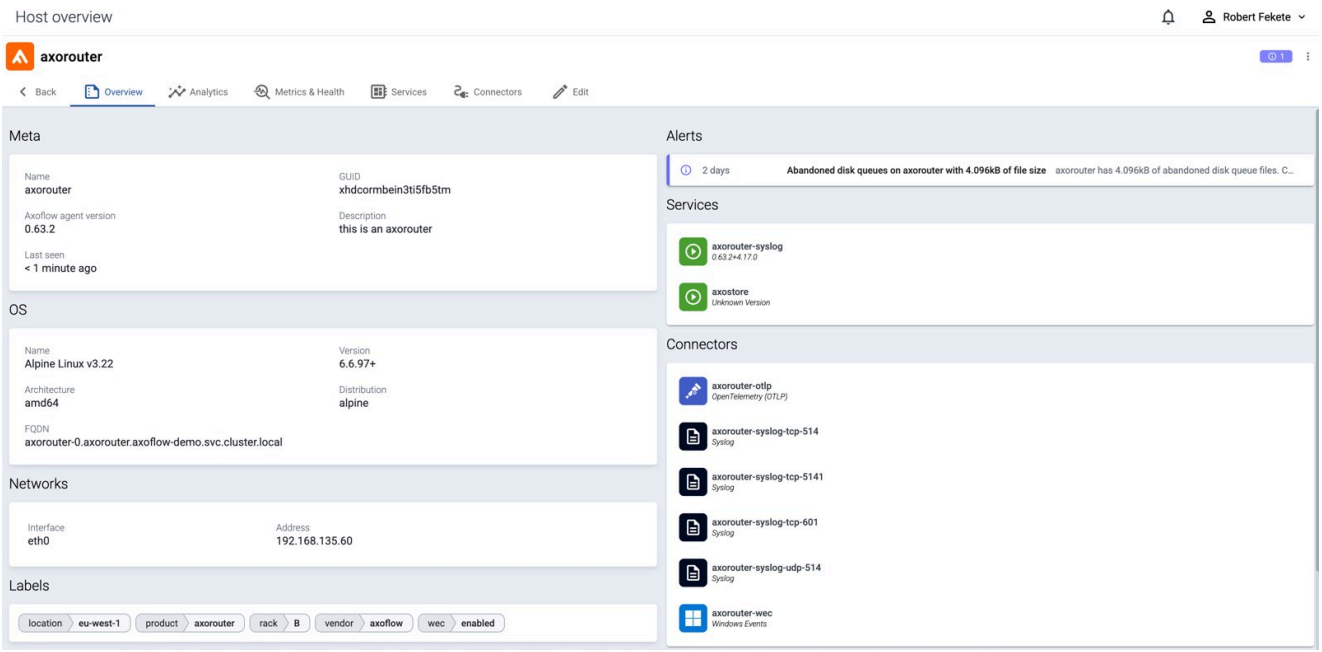
- The version on the agent
- The name and version of the log collector running on the host (for example, AxoSyslog or Splunk Connect for Syslog)
- Operating system, version, and architecture
- When AxoConsole has last received metrics from the host
- Resource information: CPU and memory usage, disk buffer usage. Click on a resource to open its resource history on the **Metrics & Health** page of the host.
- Traffic information: volume of the incoming and outgoing data traffic on the host.
- Cloud-related labels: If the host is running in the cloud, the provider, region, zone labels are automatically available.
- An indicator if there are any active alerts on the host.

For more details about the host, select the hostname, or click ⋮ **Host Options**.

The host **Overview** page gives a quick overview of the host, including:

- The **Services**.
- Any active **Alerts**.
- the **Connectors** and **Forwarding Rules** available on Axoflow agent hosts.
- the **Connectors**, **Flows**, and **Stores** available on AxoRouter hosts.

You can also access host-specific **Analytics** and **Metrics**.



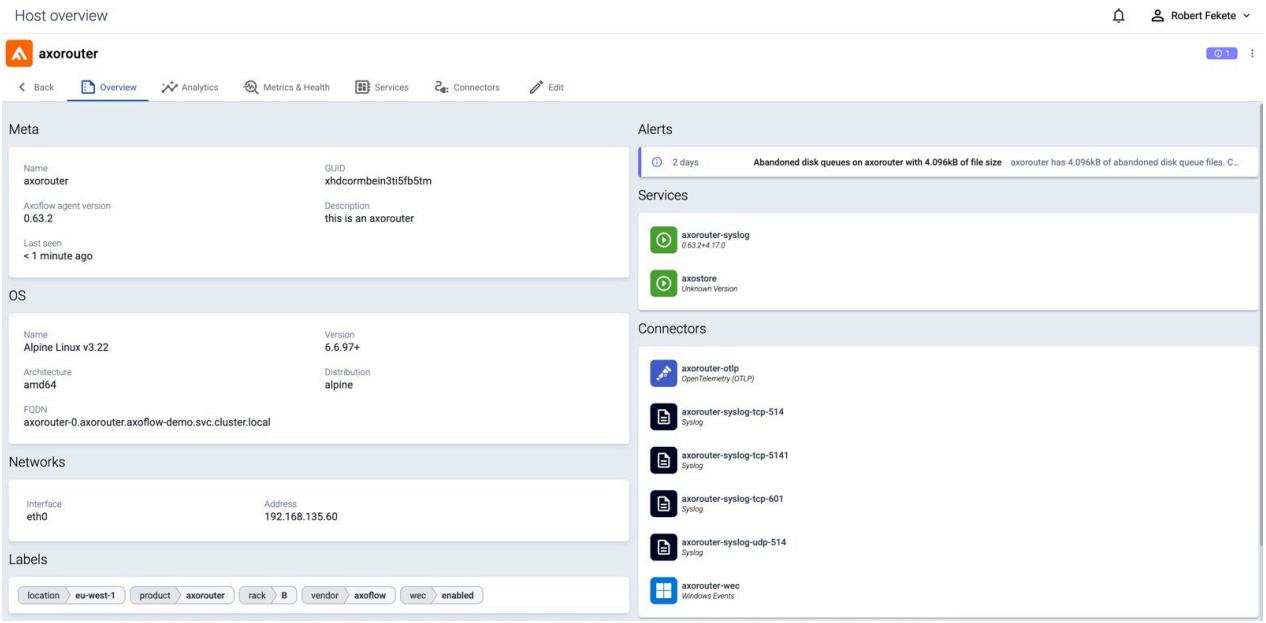
11.3.3 Custom labels and metadata

Add static custom labels to a host, so you can filter, route, and analyze its data by owner, location, or any other property.

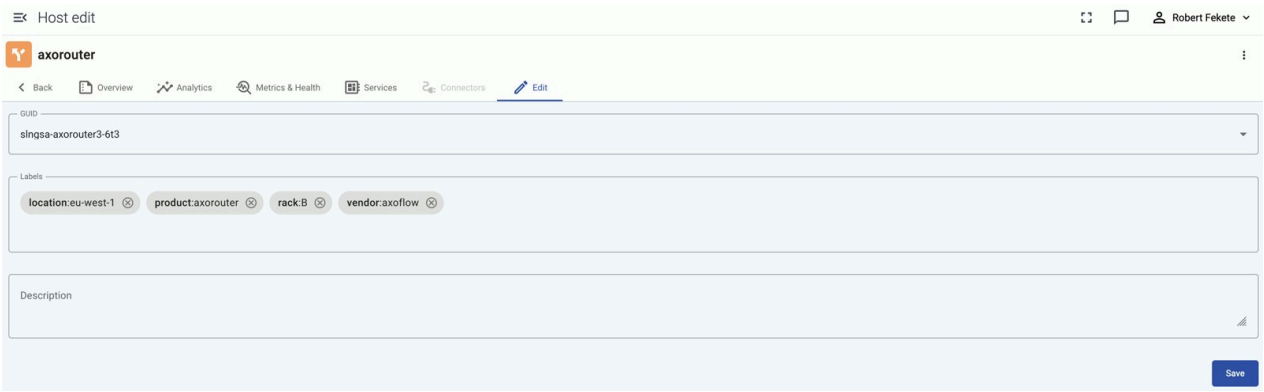
To add custom labels to a host, complete the following steps.

> **Note**
>
> Note that these are static labels.
>
> - To add labels dynamically based on the contents of the processed data, use processing steps in data flows.
> - To manage labels in bulk, see Bulk update hosts.

1. Find the host on the **Topology** page, and click on its hostname. The overview of the host is displayed. (Alternatively, you can find AxoRouters on the **Routers** page, and source and edge hosts on the **Sources** page.)



2. Select **Edit**.



You can add custom labels in `<label-name>:<value>` format (for example, the group or department a source device belongs to), or a generic description about the host. You can use the labels for quickly finding the host on the **Hosts** page, and also for filtering when configuring **Flows**.

When using labels in filters, processing steps, or search bars, note that:

- Labels added to AxoRouter hosts get the `axo_host_` prefix.
- Labels added to data sources get the `host_` prefix. For example, if you add a *rack* label to an edge host, it'll be added to the data received from the host as `host_rack`.
- Labels added on edge hosts get the `edge_connector_label_` prefix.

On other pages, like the **Overview** tab of a host, the labels are displayed without the prefixes.

3. Select **Save**.

11.3.4 Bulk update hosts

Export your host inventory to CSV, edit labels and descriptions offline, and import the file to update many hosts at once.

Instead of editing hosts one by one, you can manage your inventory in bulk: export the host list to a CSV file, edit it offline in a spreadsheet or text editor, then import the updated file. Axoflow compares the file against your current inventory and applies the changes.

Using this workflow, you can:

- Update the description and labels of existing hosts.
- Add new hosts.
- Delete hosts.

> **Note**
>
> **If you'd like to manage your hosts from an external Configuration Management Database (CMDB) like ServiceNow, contact us for integration options.**

The **Export/Import CSV** option is available on the **Routers** page (for AxoRouters) and on the **Sources** page (for source and edge hosts).

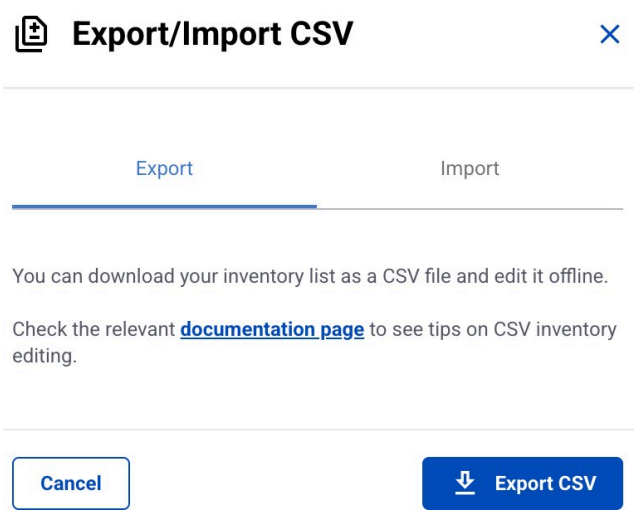
Export the inventory list

1. Open the **Routers** or the **Sources** page.
2. (Optional) Filter or search the list to narrow down the hosts you want to work with.

   > **Note**
   >
   > **The exported file contains exactly the hosts shown in the list, so any active filter or search also applies to the export.**

3. Select **Export/Import CSV**, then on the **Export** tab select **Export CSV**.

   The current list is downloaded as a CSV file.



Edit the CSV file

The exported file has one row per host and the following columns, for example:

```
name,type,description,label.app,label.building,label.discovered,label.hostGroup,lab
el.location,label.os,label.platform,label.product,label.rack,label.targetSample,lab
el.team,label.vendor,fqdn,ip
arbor-edge-ppa,APPLIANCE,,,,,,,,,arbor-edge,,,,netscout,arbor-edge-ppa,192.168.9.99
azure-us-east-2-eventhub,APPLIANCE,,,2,,8,us-east-2,,,azure-event-
hub,A,,windows,microsoft,azure-us-east-2-eventhub,10.192.23.63
```

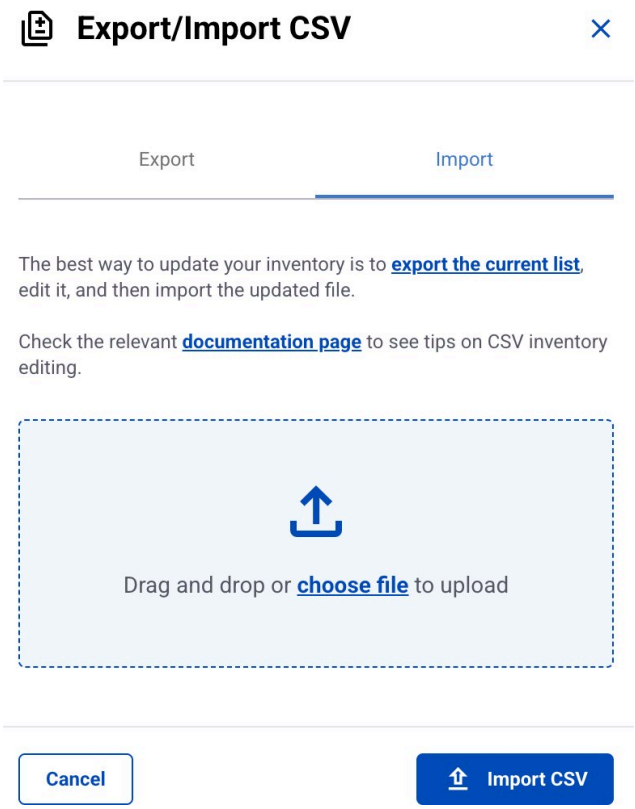
- `name` : The unique name of the host. This is what identifies the host, so keep it unchanged for hosts you want to update. The name must be a valid hostname and must be unique within the file. Required.
- `type` : The type of the host: `APPLIANCE` , `EDGE` , or `ROUTER` . You can't change the type of an existing host. To delete a host, set this column to `DELETED` .
- `description` : A free-text description of the host.
- `label.<key>` : Each label is a separate column named after its key, for example `label.location` . The cell holds the label value. To add a new label to every host, add a new `label.<key>` column. To add a label to a single host, fill in its cell and leave the others empty. Label keys follow the same rules as labels added in the UI.
- `fqdn` , `ip` : The fully qualified domain name and IP address of the host. You can only set these for `APPLIANCE` hosts. For other host types, these values are reported by the agent and can't be changed, so leave them as exported.

When editing the file, keep the following in mind:

- **The file is the desired state.** For the fields you can edit (description, labels, and the `fqdn` and `ip` of appliances), clearing a cell clears that value on the host. For example, emptying a `label.<key>` cell removes that label from the host.
- **Adding a host:** Add a new row. You can only add type `APPLIANCE` hosts, and each new host needs at least an `fqdn` or an `ip`. To add a ROUTER or an EDGE host, provision an AxoRouter or an Axoflow agent.
- **Deleting a host:** Set the `type` column of the row to `DELETED`.
- **Hosts you leave out are untouched.** Only the rows in the file are processed. Removing a row from the file does not delete the host – use `DELETED` for that.

Import the edited CSV file

1. On the **Routers** or the **Sources** page, select **Export/Import CSV**, then open the **Import** tab.

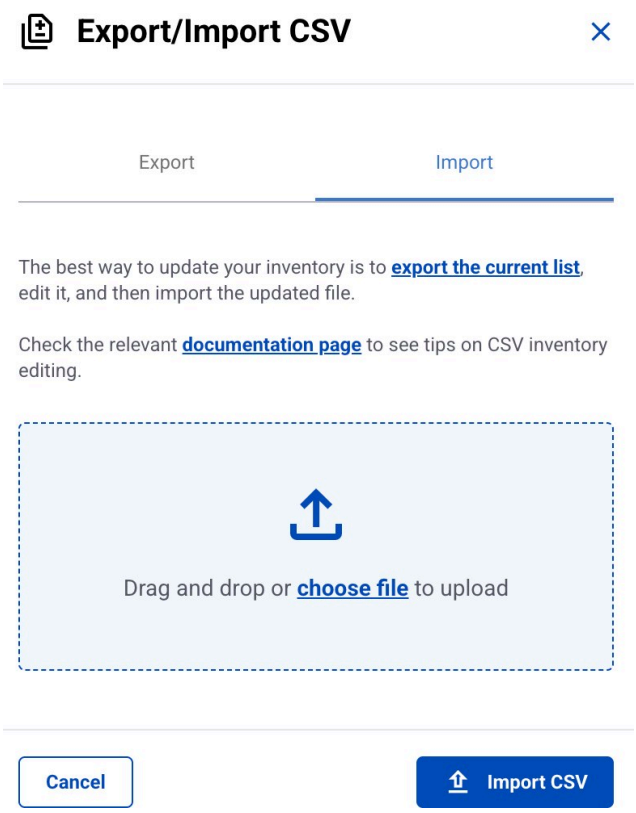


2. Upload the edited CSV file.

   AxoConsole compares the file against the current inventory and shows a summary of the changes to be made: how many hosts will be added, updated, and removed.

   If the file contains errors, they're listed per row and column so you can correct them and upload the file again.

3. Review the summary.



Note

Changes are applied one by one, and are listed one-by-one in the Activity Logs. To stop an import that's in progress, close the dialog. Changes that have already been applied stay in place.

CAUTION:

Deleting hosts (`DELETED`) and clearing values can't be undone. Review the change summary carefully before importing.

Select **Import CSV** to apply the changes.

11.3.5 Services

Check which collector and router services run on a managed host, and their versions.

The **Services** page of an AxoRouter or edge host shows information about the data collector or router services running on the host. This page is only available for managed pipeline elements.

The following information is displayed:

- **Name**: Name of the service
- **Version**: Version number of the service
- **Type**: Type of the service (which data collector or processor it's running), for example, `axorouter-syslog`, `axorouter-wec`, or `axostore`
- **Supervisor**: The type of the supervisor process. For example, Splunk Connect for Syslog (`sc4s`) runs a syslog-ng process under the hood.

Icons and colors indicate the status of the service: `running`, `stopped`, or `not registered`.

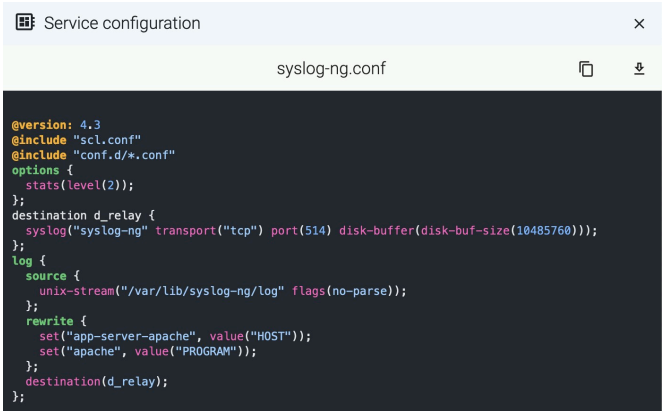


Depending on the configuration of the host, the following services can be available:

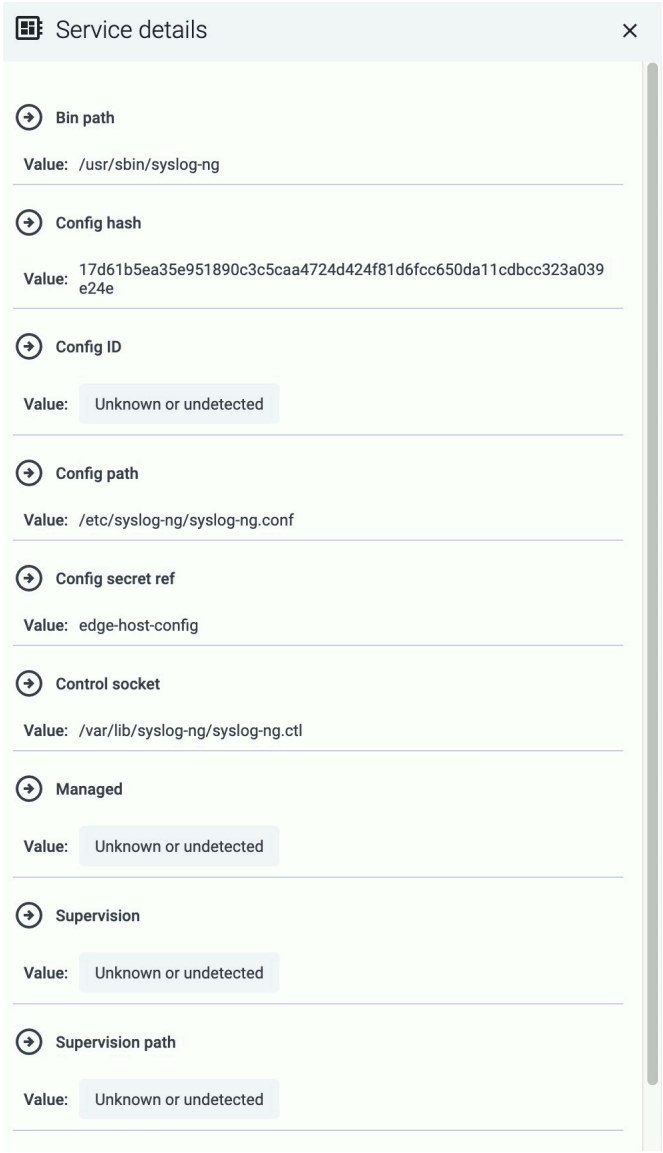
- **axoflow-cloudconnector**: The service that collects logs from cloud services and APIs.
- **axoflow-otel-collector**: The collector agent on edge hosts.
- **axorouter-logstore**: The service that handles Logstore type stores.
- **axolet**: The monitoring and management agent for Axoflow pipeline elements.
- **axorouter-resolver**: A service for retrieving host data from external CMDB systems.
- **axorouter-syslog**: The main processing element of AxoRouter deployments.
- **axorouter-wec**: The service that handles the Windows Events connector (WEC).
- **axostore**: The service that handles AxoStore on the host. Only available if there are stores configured on the host.

Service configuration

To check the configuration of the service, select **Configuration**. This shows the list of related environment variables and configuration files. Select a file or environment variable to display its value. For example:



To display other details of the service (for example, the location of the configuration file or the binary), select **Details**.



Manage service

- To reload a registered service, select **Reload**. Reloading applies the service's configuration without restarting it, so the service keeps running.
- To restart a registered service, select **Restart**. Restarting stops and starts the service completely, so it will be unavailable for a short time.

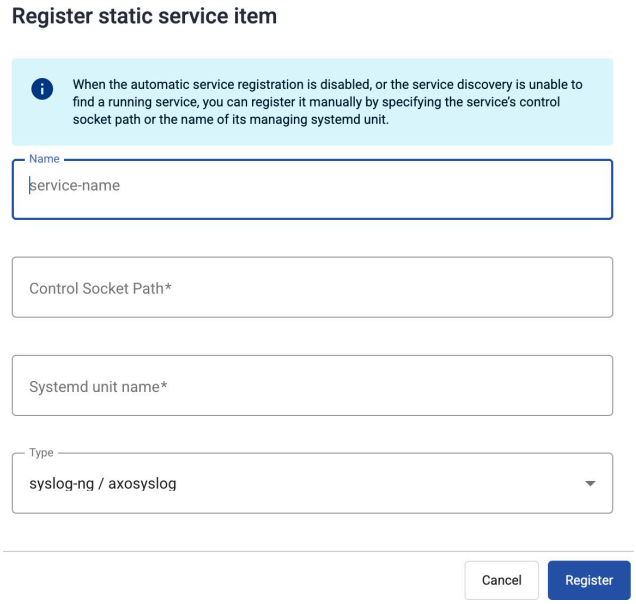
Register service manually

If **Automatic service registration** is disabled, or the service discovery is unable to find a running service, you can register it manually by specifying:

- the service's control socket path, or
- the name of its managing systemd unit.

To register a service manually, complete the following steps.

1. Find the host and open its **Services** page.
2. Select **Register static service**.
3. Enter a name for the service. This name will be displayed in the services list.



4. Provide at least one of the following:
   - **Control Socket Path**: The path to the Unix domain socket used to communicate with the service, for example, `/var/lib/syslog-ng/syslog-ng.ctl`.
   - **Systemd unit name**: The name of the systemd service of the service you're registering, for example, `syslog-ng.service`.
5. Select **Register**.

11.4 Log tapping

Sample live messages on a host, filter them by label, and compare the original message with what Axoflow sends to the destination.

Log tapping in Axoflow samples the log flow. You can use labels to filter for specific messages (like ones with parse errors) and tap only those messages. To not get overwhelmed with events, Axoflow automatically samples the output: if many messages match the selected filter, only a subset is shown (about 1 message per second). Using log tapping, you can quickly troubleshoot both parsing/curation errors and destination ingest (API) errors, and check:

- What was in the original message?
- What is sent in the final payload to the destination?

To see log tapping in action, check this blog post.

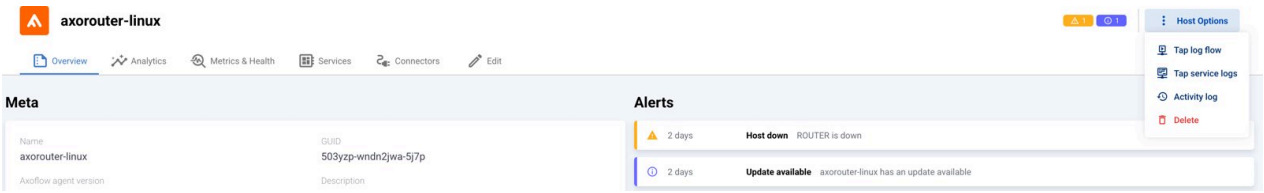
> **Note**
>
> **Neither AxoRouter nor AxoConsole records any log data. The Flow tapping and Log tapping features only send a sample of the data to your browser using encrypted connections while the tapping is active. Only authorized users can use tapping.**

- Tap into your log flow.
- Display the logs of the log collector service. Service logs also contain the output of the systemd units used to launch the containers (where applicable).

Tap into the log flow

To tap into your log flow, complete the following steps.

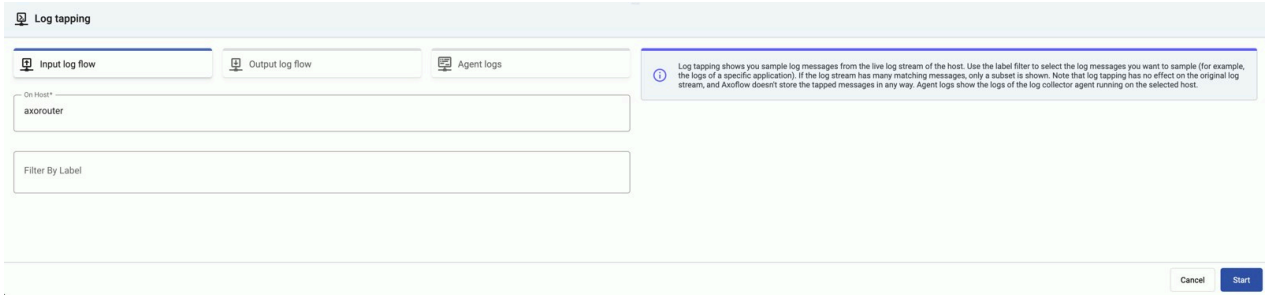
1. Select the AxoRouter where you want to tap the logs on the **Topology** or **Routers** page. Alternatively, select `⌘/Ctrl + K` and enter the name of the AxoRouter.
2. Select ⋮ **Host Options > Tap log flow**.



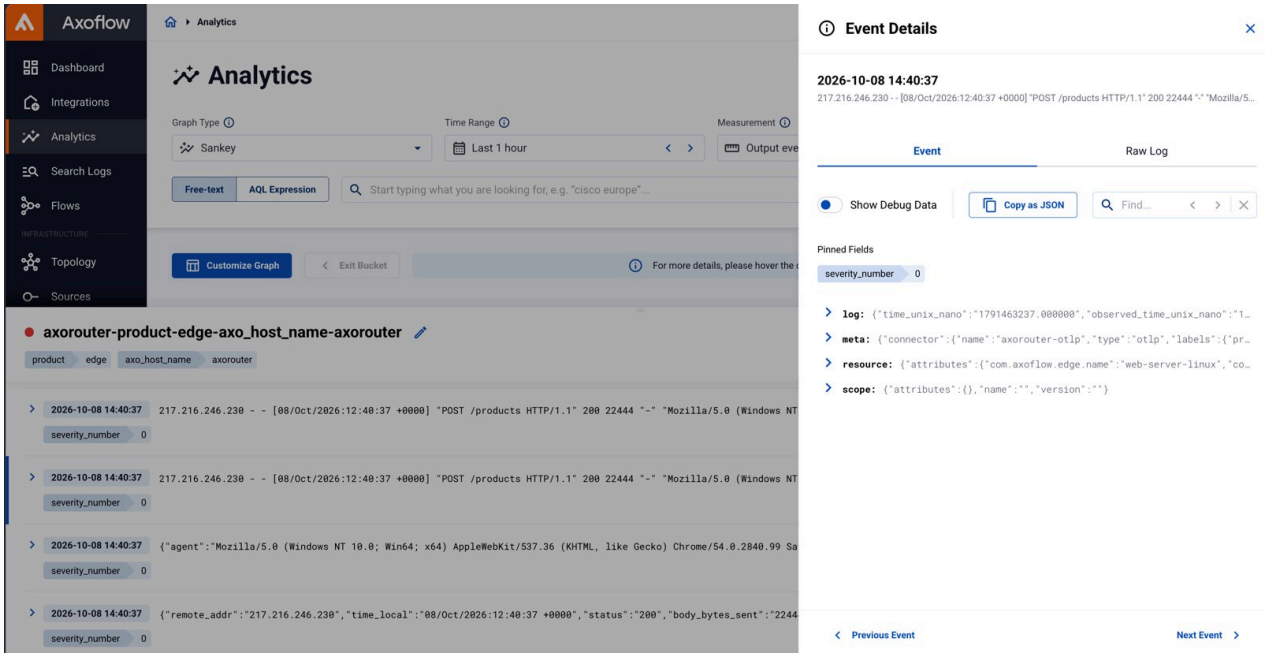
3. Tap into the log flow.
    - To see the input data, select **Input log flow > Start**.

- To see the output data, select **Output log flow > Start**.

You can use labels to filter the messages and sample only the matching ones.

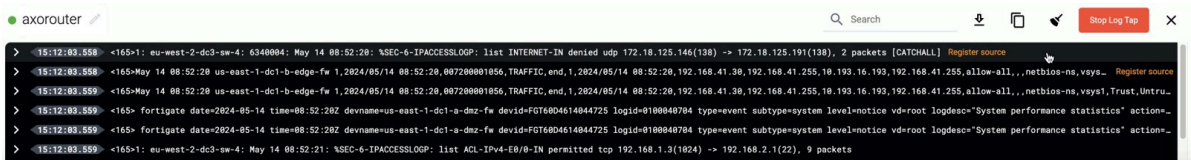


4. When the logs you're interested in show up, click **Pause**, then click the info icon next to a log message to see its details. For details about the message schema and the displayed fields, see Message schema reference. To also display AxoRouter's internal representation of the message, see Show debug data.



Note

For sources that are not yet registered in the AxoConsole host database, the **Register source** button is shown at the end of the message. Click it to add the source to AxoConsole: this will allow attributing the logs coming from the given source as such, enriching the messages, the analytics data, as well as the **Topology** page.



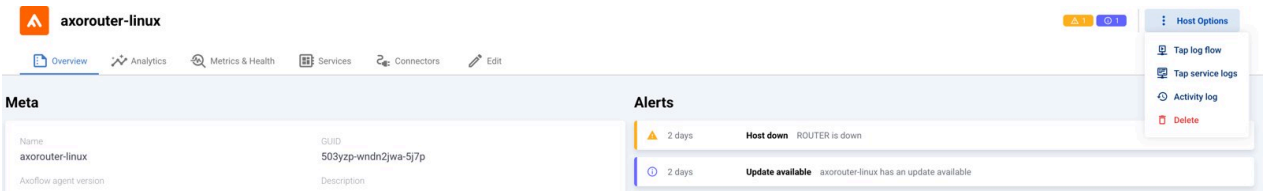
> **Note**
>
> **When using Log tapping, ETW events look a bit weird: the body of these events is empty. That's normal, the reason for that is that everything is sent as metadata.**

Tap service logs

Display the logs of the log collector service, complete the following steps. AxoConsole supports service log tapping on AxoRouter and Axoflow agent for Linux hosts.

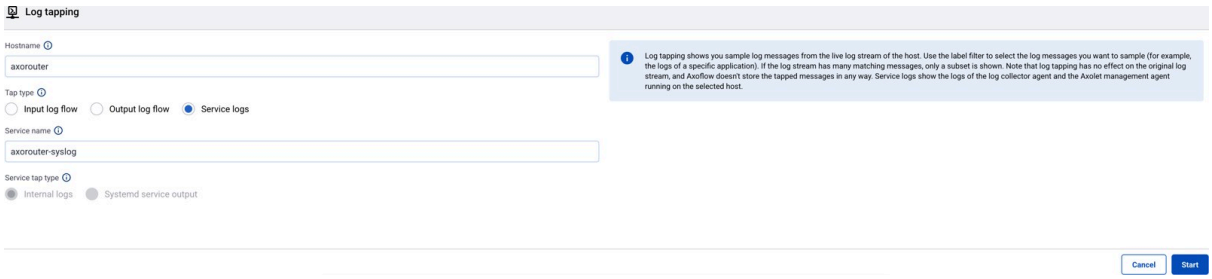
1. Find the host you want to tap.
   - Select the AxoRouter on the **Topology** or **Routers** page.
   - Select the Axoflow agent on the **Topology** or **Sources** page.

   Alternatively, select `⌘/Ctrl + K` and enter the name of the host.
2. To start log tapping instantly, select **Services** > ✓ **Service log** in the row of the `axorouter-syslog` or `axoflow-otel-collector.service` service.



   To filter the logs or to tap the logs of a specific service, select ⋮ **Host Options** > **Tap service logs** and complete the following steps.

   1. Select **Service logs**.



   2. Set the **Service name** field to see the logs of a specific service.

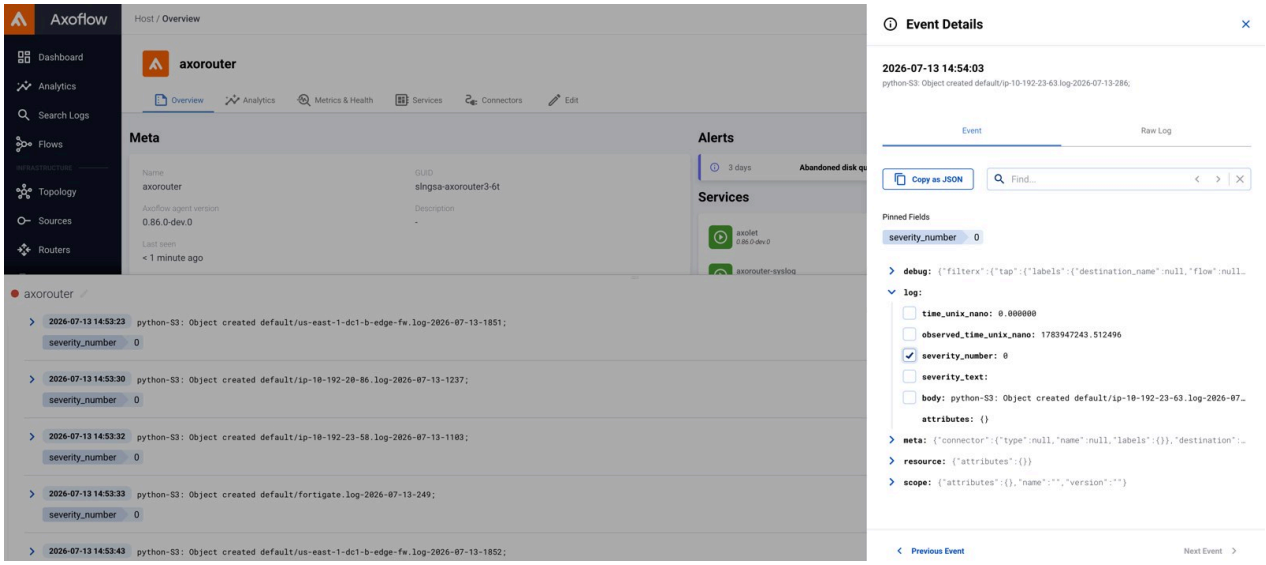
      Depending on the configuration of the host, the following services can be available:

      - **axoflow-cloudconnector**: The service that collects logs from cloud services and APIs.
      - **axoflow-otel-collector**: The collector agent on edge hosts.

- **axorouter-logstore**: The service that handles Logstore type stores.
- **axolet**: The monitoring and management agent for Axoflow pipeline elements.
- **axorouter-resolver**: A service for retrieving host data from external CMDB systems.
- **axorouter-syslog**: The main processing element of AxoRouter deployments.
- **axorouter-wec**: The service that handles the Windows Events connector (WEC).
- **axostore**: The service that handles AxoStore on the host. Only available if there are stores configured on the host.

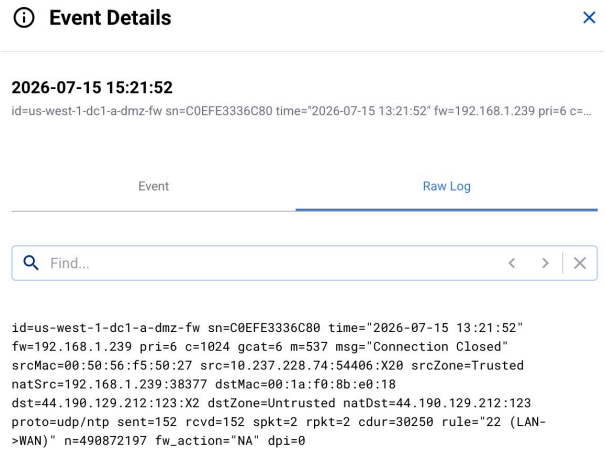
If systemd is available on the host, select **Systemd service output** to show the logs of the service from the systemd journal. Otherwise, the **Internal logs** of the service are available.

3. When the logs you're interested in show up, click **Pause**, then click the info icon next to a log message to see its details.

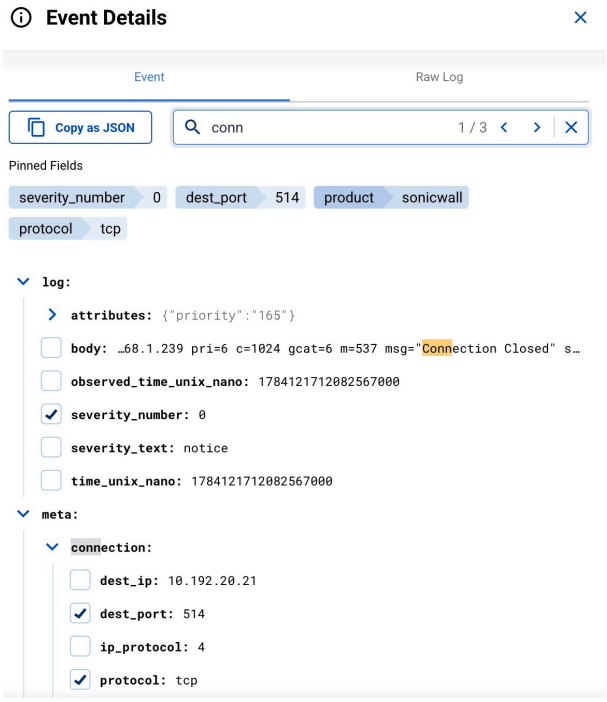


The **Event Details** panel allows you to inspect a specific event. For details on the available fields and their structure, see Message schema reference.

- To display the original log message, select **Raw Log**.

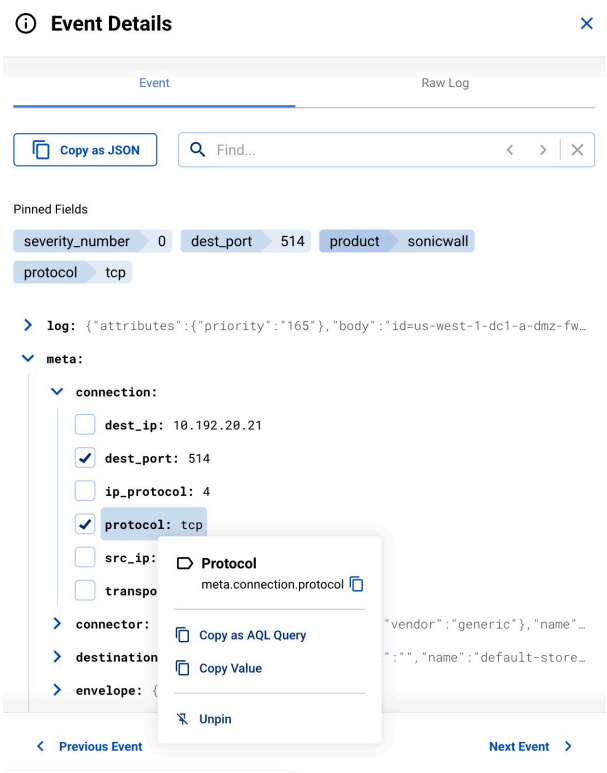


- To copy the entire event as JSON to the clipboard, select **Copy as JSON**.
- To search the event body and metadata, enter a search keyword into the **Find** field. This searches for the string on the **Event Details** panel, including the log, metadata, or label names and values.



- To add a field to the **Pinned Fields** and display it for every event, click the checkbox before the field. Pinned fields show up in the event details, and also in the list of the events, making it easy to get a quick glance on the distribution of the field values. They're included as separate fields when you **Export** the search results into a CSV file.

  To unpin a field, select the field in any of the events, then select **Unpin** from the local menu.



- To copy the name or the value of a field, select the name of the field, then select content_copy or **Copy Value**.
- To copy an AQL query that matches this field and its current value, select the name of the field, then select **Copy as AQL Query**. You can paste this query into the **AQL** search field, and use it for filtering.
- If the field refers to a pipeline element in Axoflow (for example, a flow in the `meta.flow` field, a store, or an AxoRouter), you can jump to the page of the related element.

Service status messages

While a tapping session is active, AxoConsole shows status messages about the tapped service inline in the log stream. They appear in every pane, regardless of the tapping point of the pane. These messages don't come from your data sources. Axoflow inserts them to tell you when the service is restarted or its configuration is reloaded, for example, after you save a change to a flow or a connector rule. That way you can tell whether a change in the messages you see is the result of your configuration change, and notice when applying the new configuration failed.

The messages are color-coded by severity: informational messages (`notice` severity) are blue, errors (`err` severity) are red.

The following status messages can appear:

Message	Severity	Meaning
`Restarting service`	`notice`	AxoConsole has started restarting the service.
`Force restarting service`	`notice`	AxoConsole has started a forced restart of the service (the **Force restart** option of the **Restart** dialog on the **Services** tab of the host). A forced restart terminates the service without waiting for the delivery of in-flight messages.
`Service restart initiated successfully`	`notice`	The service has accepted the restart request.
`Restart failed: <details>`	`err`	The service couldn't be restarted.
`Reload complete`	`notice`	The service is running with the new configuration.
`Reload failed: <details>`	`err`	The service couldn't apply the new configuration, and keeps running with the previous one. Usually this means that the new configuration is invalid, for example, a FilterX processing step contains a syntax error.
`Reload blocked due to missing config files: <details>`	`err`	The service couldn't apply the new configuration because a file it references (such as a certificate) isn't available on the host.

Show debug data

The details of a tapped message can include a `debug` field that shows how AxoRouter represents the message internally:

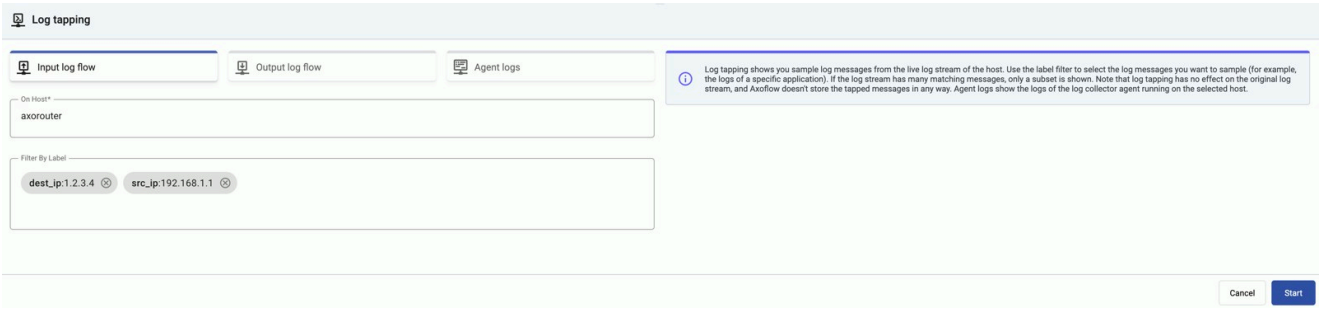
- `debug.filterx` contains the FilterX variables that are set when the message reaches the tapping point.
- `debug.msg` contains the name-value pairs of the message.

These fields are hidden by default. To display them, open the details of a message, then select **Show Debug Data**. The switch is available only for messages that have debug data.

Debug data is useful when you're troubleshooting a processing step that doesn't behave as you expect, or when Axoflow support asks you for it. For the fields of the message itself, see Message schema reference.

Filter the messages

You can add labels to the **Filter labels** field to sample only messages matching the filter. If you specify multiple labels, only messages that match all filters will be sampled. For example, the following filter selects messages from a specific source IP, sent to a specific destination IP.



For details about the labels available for filtering, see Metrics schema. To tap the messages received from edge hosts, you can use the related metrics labels of the edge collector, for example, `edge_connector_type:windowsEventLog` samples only the event log messages received from edge hosts. For details about the message schema and the available fields, see Message schema reference.

11.5 Alerts

Find, filter, and read the alerts AxoConsole raises about unavailable destinations, dropped messages, full disk queues, and other problems.

Axoflow raises alerts for a number of events in your security data pipeline, for example:

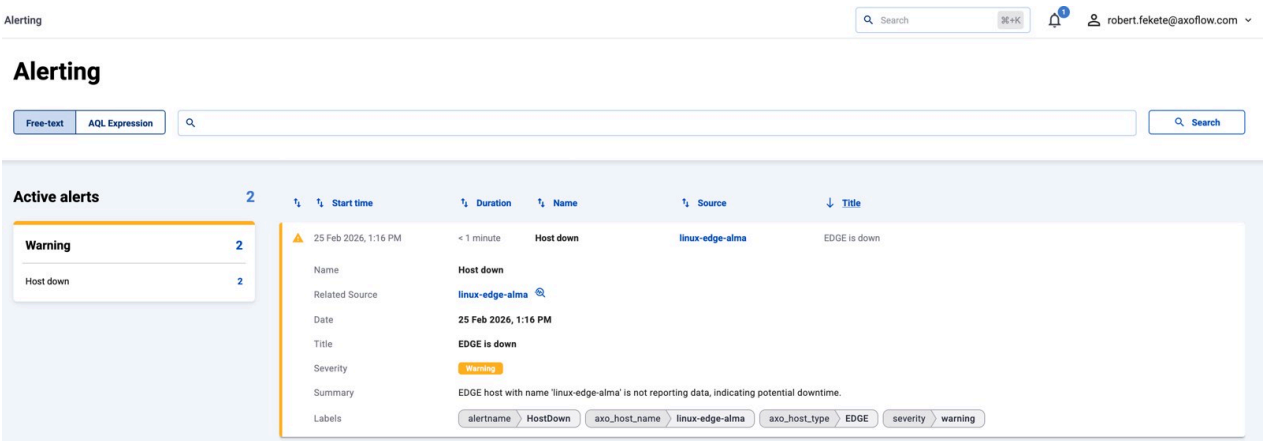
- a destination becomes unavailable or isn't accepting messages,
- a host is dropping packets or messages,
- a host becomes unavailable,
- the disk queues of a host are filling up,
- abandoned (orphaned) disk buffer files are on the host,
- the configuration of a managed host failed to synchronize or reload,
- traffic of a flow has unexpectedly dropped,
- a new AxoRouter or Axolet version is available.

> **CAUTION:**
>
> AxoConsole can automatically raise an alert if a source that used to send data regularly stops sending data. To enable this alert for a source, add the `traffic_alerts:true` label to the source.

For the details of specific alerts, see Alerts reference.

Alerts are indicated at a number of places:

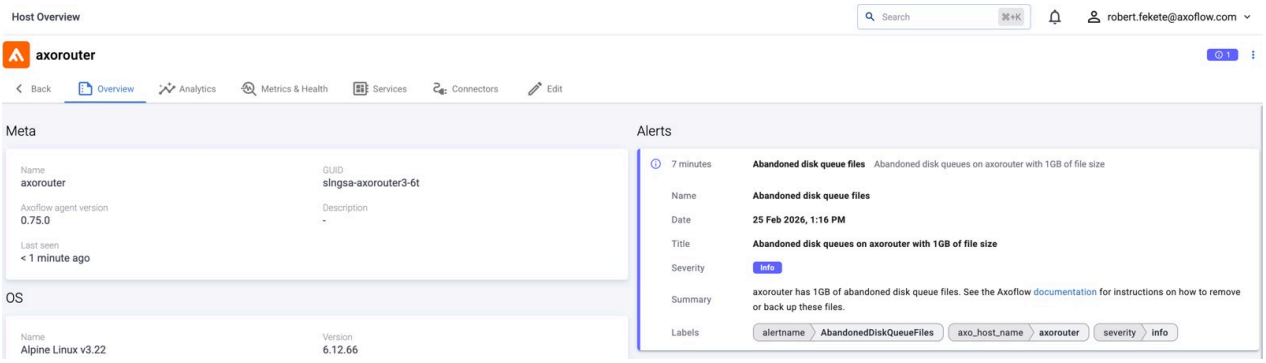
- On the **Alerting** page. Select an alert to display its details.



- On the **Topology** page:



- On the **Overview** page of the host:



- On the **Metrics & Health** page of the host the alerts are shown as an overlay to the metrics:



Filter alerts

You can use the Filter Bar on the **Alerts** page to search and filter for specific alerts.

The settings of the filter bar change the URL parameters of the page, so you can bookmark it, or share a specific view by sharing the URL.

- **Free-text** mode searches in the following fields of the alerts: **Id**, **Name**, **Title**, **Summary**, **Labels**.

  The search is case insensitive. Adding multiple keywords searches for matches in any of the previous fields. This is equivalent to the `@ANY =* keyword1 AND @ANY =* keyword2` AQL query.

- **AQL Expression** mode allows you to search in specific fields of the activity logs.

It also makes more complex filtering possible, using the Equals, Contains (partial match), and Match (regular expression match) operators. Note that:

- To execute the search, click **Search**, or hit ESC then ENTER.
- AxoConsole autocompletes the built-in and custom labels and field names, as well as their most frequent values, but doesn't autocomplete labels and variables created by data parsing and processing steps.
- You can use the AND and OR operators to combine expressions, and also parenthesis if needed. For details on AQL, see AQL operator reference.
- The precedence of the operators is the following: parentheses, AND, OR, comparison operators.
- Use the usual keyboard shortcuts to undo (⌘/Ctrl + Z) or redo (⌘/Ctrl + Shift + Z) your edits.

Alerts have the following fields:

- **Name**: The name of the alert, for example, `Target down`.
- **Related Source**: The host where the alert event occurred.
- **Date**: The time when the alert triggered.
- **Title**: The title of the alert, for example, `<host> is down`.
- **Severity**: The severity level of the alert: `Info`, `Warning`, or `Critical`.
- **Summary**: A detailed description of the alert.
- **Duration**: How long the alert lasted.
- **Labels**: Labels related to the alert.

Troubleshooting

If the description of the alert doesn't give instructions on solving the issue, see Troubleshooting Axoflow.

11.5.1 Alerts reference

Look up each alert that AxoConsole can raise about your pipeline, with its severity and what triggers it.

Host down (warning)

`host-type` host with name '`hostname`' isn't reporting data, indicating potential downtime.

An edge or router host has stopped reporting data to the AxoConsole. Because the system tolerates short reporting gaps, the alert takes around seven minutes to fire after the host actually goes silent. The usual causes are loss of network connectivity from the host, Axolet being stopped on the host, or a hardware or operating-system failure on the host itself.

Related metrics: cpu, memory

Host clock skew (warning)

The clock on host '`hostname`' differs from the AxoConsole clock. A wrong host clock causes trouble in several places, for example AxoRouter can no longer work out the timezone of an appliance that sends timestamps without one. Enable NTP on the host.

The value is the AxoConsole clock minus the timestamp the host put on its metrics push. A positive value means the host clock is behind, a negative one means it is ahead. The time a sample spends in the push queue and on the wire only ever adds to the value, which is why the bounds differ: `-10s` and `+60s`. For the same reason the alert reads the lowest sample in a 30 minute window, the one that spent least time in transit.

Related metrics: eventDelaySeconds

Cert Expires (info)

`host-type` host with name '`hostname`' uses a certificate chain that expires in less than 45 days. It should have been renewed automatically.

The client certificate chain that an edge or router host uses to authenticate to the AxoConsole will expire in less than 45 days. This covers the host's own certificate and the intermediate CA it was issued under. Certificates should be renewed automatically; if this alert appears, automatic renewal has failed. Restarting the `axolet` service on the affected host often triggers another renewal attempt. If the alert doesn't clear within five minutes, contact Axoflow support.

Stats level (info)

'`hostname`' is configured with a stats level of 0.

An `axorouter-syslog` (or another AxoSyslog-based) service on a host is configured with a stats level of 0, so most metrics shown on the AxoConsole will be incomplete or missing for it. The recommended stats level is 2. Add `options { stats(level(2)); };` to the top of the main configuration file and reload the service. See the AxoSyslog stats option reference.

Metric series leak (warning)

Most classification metric series on '`hostname`' have counted a single message, so a source creates a new series for nearly every message it receives. The growing series count slows down metrics collection from the host, leaves gaps in its metrics, and can trigger false alerts. Check the sources of this host for a sender whose messages don't match the format of its connector.

Counts the `syslogng_classified_events_total` series of a host that have counted exactly one message. A labeled field (program name, hostname, source IP) that takes a new value for nearly every message creates one series per message, and the series exist until AxoRouter restarts. A typical cause is a sender that uses a different protocol than its connector, for example, Beats (Lumberjack) sent to a syslog port: the syslog parser takes a random token of each fragment as the program name.

Related metrics: connections

High CPU (warning)

CPU usage at `hostname` has been above 90% for the last 15 minutes.

The average CPU usage across all cores on a host has been above 90% for at least 15 minutes. Sustained high CPU usage can prevent the host from keeping up with its workload, and may lead to message drops, packet drops, or processing delays.

Related metrics: cpu, droppedPacketsTotal, logInputEvents

Message drop (warning)

Messages have been dropped on `hostname` in the last 10 minutes.

An `axorouter-syslog` service on a host has dropped messages in the last 10 minutes. This happens when a destination can't accept messages fast enough and flow control isn't configured to back-pressure the source. Once the destination's output buffer overflows, further incoming messages are discarded and can't be recovered.

Related metrics: droppedPacketsTotal, logInputEvents, logOutputEvents

UDP packet drop (warning)

UDP packets have been dropped on `hostname` in the last 10 minutes.

A host has dropped UDP packets in the last 10 minutes. The most common cause is that the kernel's UDP receive buffer is full: packets that arrive while the buffer is saturated are silently discarded by the kernel before the service can read them. An undersized socket buffer on the receiving connector is a frequent contributor (see also the `AxorouterLowRmemMax` alert).

Related metrics: droppedPacketsTotal, logInputEvents, logOutputEvents

Collector send failure (warning)

`service-id` on `hostname` failed to send log records to its `exporter` destination. Check the destination's reachability and the service's logs on the host.

A collector service on a host (a cloud connector or the OpenTelemetry collector) failed to deliver log records to its configured destination in the last 10 minutes. For a cloud connector, this destination is the AxoRouter that receives its cloud logs, so the logs collected from the cloud source don't reach the log pipeline. The alert fires per exporter, so a service with several failing destinations produces several concurrent alerts. The service retries records from the export queue while it has room, and drops them once the queue is full. Common causes include a destination endpoint that is unreachable or misconfigured, and a certificate that was rejected or has expired.

Related metrics: droppedPacketsTotal, logOutputEvents, logQueue

Collector receive failure (warning)

`service-id` on `hostname` refused log records from its `receiver` source. Check the source's configuration and the service's logs on the host.

A collector service on a host (a cloud connector or the OpenTelemetry collector) refused log records from one of its sources in the last 10 minutes. The alert fires per receiver, so a service with several failing sources produces several concurrent alerts. The refused records are lost unless the source itself retains and resends them. For a cloud connector reading Azure Event Hubs, this usually means that the records couldn't be read as logs, or that the event hub no longer exists or isn't reachable with the configured connection string. Check that the event hub still exists, that its connection string is valid, and that it carries the log format the connector expects.

Related metrics: logInputEvents, logInputBytes

Abandoned disk queue files (info)

There are abandoned disk queues on `hostname`. See the Axoflow documentation for instructions on how to remove or back up these files.

A disk queue file on a host is no longer attached to any running `axorouter-syslog` instance. Such files occupy disk space and, if they contain messages, those messages remain stranded until they're manually recovered or wiped. Use `dqtool cat <filename>` to inspect a file before removing it; see the `axorouter-ctl wipe` documentation for the supported procedure.

Related metrics: logDiskQueue, logDiskQueueBytes

Abandoned messages (warning)

There are messages abandoned in disk queues. See the Axoflow documentation for instructions on how to remove or back up these files.

A disk queue file on a host is no longer attached to any running `axorouter-syslog` instance, and that file still contains messages. Those messages remain stranded until they're manually recovered or the file is wiped. Use `dqtool cat <filename>` to inspect a file before removing it; see the `axorouter-ctl wipe` documentation for the supported procedure.

Related metrics: logDiskQueue, logDiskQueueBytes

Destination unreachable (warning)

Unreachable `driver` destinations. Check network connectivity and agent logs.

The `axorouter-syslog` service on a host has been unable to reach a configured destination for at least five minutes. The alert fires per destination driver and address, so a single host with multiple broken destinations produces multiple concurrent alerts. The service buffers messages for the unreachable destination in memory or on disk. AxoRouter may drop further messages once buffers fill up. Check the destination's network reachability and the service's logs on the host.

Related metrics: droppedPacketsTotal, logMemoryQueue, logDiskQueue, logMemoryQueueBytes, logDiskQueueBytes, eventDelaySeconds

Timestamp timezone error (warning)

Messages that a flow routes to its destination carry timestamps that are offset by a whole number of hours. AxoRouter cannot determine the timezone of the sending appliance, so the destination indexes these messages under the wrong time and a search over a time range misses them. Check the analytics breakdown of the flow to see which appliances produce it and what share of the traffic is affected, then set the correct timezone on the appliance, or add a Fix Timestamp processing step that names its timezone. AxoRouter only labels an event when it can resolve the offset, so more messages may be affected than this alert counts.

AxoRouter labels an output event with `suspected_time_error_hours` when `guess_timezone()` moves the in-message timestamp. That function compares the timestamp against the current time, rounds the difference to the nearest quarter hour, and only accepts a residual of 30 seconds or less that maps to a real timezone offset. The label holds the signed error in hours: `+2` means the timestamp reads two hours later than the true time, `-5.5` means five and a half hours earlier. Only differences that map to a real timezone offset are labelled, so an ordinary clock skew of a few seconds or minutes produces no alert. The alert fires per flow, destination and offset value, so one flow fed by appliances in two wrong timezones produces two alerts. The `flow` label makes the Console attach the alert to that flow. A connector that pulls batched data from a cloud API can also produce this label, because the reference is the current time rather than the time the batch was pulled.

Related metrics: logOutputEvents, eventDelaySeconds

HTTP 5xx (warning)

High rate of HTTP `response-code` on `hostname` to `url`

HTTP requests sent to a destination URL have been returning 5xx responses in the last minute, and no 2xx responses have been observed for the same URL in that window. The alert fires once per destination URL and response code, so a destination returning several distinct 5xx codes can produce several alerts. A 5xx response typically points to a server-side problem (at the destination, or in a proxy in front of it) rather than to a sender-side configuration mistake.

Related metrics: droppedPacketsTotal, logOutputEvents, logDiskQueueBytes

HTTP 4xx (warning)

Increased rate of HTTP `response-code` on `hostname` to `url`

HTTP requests sent to a destination URL have been returning 4xx responses in the last minute, and no 2xx responses have been observed for the same URL in that window. The alert fires once per destination URL and response code, so a destination returning several distinct 4xx codes can produce several alerts. A 4xx response usually indicates a problem at the sender side rather than a fault at the destination: invalid credentials, missing permissions, a target resource (bucket, stream, topic) that doesn't exist at the destination, or a malformed request.

Related metrics: droppedPacketsTotal, logOutputEvents, logDiskQueueBytes

gRPC error (warning)

Increased rate of gRPC `response-code` responses on `hostname` to `driver`

gRPC requests sent to a destination have been returning non-`ok` status codes in the last minute. The alert fires once per destination URL, driver, topic, and response code. Common causes include misconfigured credentials, missing permissions, or a target resource (bucket, stream, topic) that doesn't exist at the destination.

Related metrics: droppedPacketsTotal, logOutputEvents, logDiskQueueBytes

Buffer filling up (warning)

Disk buffer `path` is predicted to fill up within 10 minutes on `hostname`

The remaining capacity of a disk buffer on a host is trending down fast enough that, at the current rate, it's predicted to fill up within the next 10 minutes. The alert fires per disk buffer path, so a host with multiple buffers can produce multiple alerts. Once a disk

buffer fills up entirely, the service stops being able to enqueue further messages for the affected destination and may start dropping them.

Related metrics: logDiskQueueBytes, networkInputBytes

Buffer is full (critical)

Disk buffer `path` is filled up on `hostname`

A disk buffer on a host has filled up: the service can't enqueue further messages for the affected destination. Messages may be dropped until the buffer drains enough to accept new writes, which requires either the destination to start accepting traffic again or the buffer to be enlarged.

Related metrics: logDiskQueueBytes, networkInputBytes

Config Sync Error (warning)

An error occurred on host `hostname` during configuration synchronization for service `service-id`. Check the `axolet` logs for details.

Axolet on a host has been unable to write a new configuration to disk for at least five minutes. The previously synchronized configuration remains in place, so the managed services on the host keep processing traffic, but changes made in the AxoConsole won't take effect on this host until synchronization succeeds again. Common causes include a filesystem permissions issue or a missing target directory on the host.

Config Load Error (warning)

Agent configuration load failed on `hostname` for service `service-id`. Check the `axolet` logs for details.

An `axorouter-syslog` service on a host failed to load a new configuration. The previously loaded configuration remains active, so traffic processing continues, but changes made in the AxoConsole won't take effect on this host until the load succeeds. Common causes include a syntactically invalid generated configuration, or a missing file on the host (see also the `ServiceMissingFile` alert).

WEC Config Generation Error (warning)

WEC configuration generation failed on '`hostname`' for service '`service-id`'. Contact Axoflow support.

Configuration generation for the Windows Event Collector service on a host has been failing for at least five minutes. The host continues to use the last successfully generated WEC configuration, but changes made in the AxoConsole don't reach this host until generation succeeds again. This typically indicates an internal issue. Contact Axoflow support.

AxoStore Config Generation Error (warning)

AxoStore configuration generation failed on '`hostname`' for service '`service-id`'. Contact Axoflow support.

Configuration generation for the AxoStore service on a host has been failing for at least five minutes. The host continues to use the last successfully generated AxoStore configuration, but changes made in the AxoConsole don't reach this host until generation succeeds again. This typically indicates an internal issue. Contact Axoflow support.

Cloud connector config generation error (warning)

Cloud connector configuration generation failed on '`hostname`' for service '`service-id`'. Contact Axoflow support.

Configuration generation for the cloud connector service on a host has been failing for at least five minutes. The host continues to use the last successfully generated cloud connector configuration, so cloud sources added or changed in the AxoConsole don't reach this host until generation succeeds again. This typically indicates an internal issue. Contact Axoflow support.

Flow stuck in error state (warning)

Unable to provision paths for Flow `flow` due to error state. Check AxoConsole for details.

A Flow is in an error state, so the system can't provision its data paths. The Flow's status on the AxoConsole contains the underlying reason. Until the error is resolved, traffic that would have flowed through this Flow isn't processed.

Flow has no traffic (warning)

Flow `flow` has no traffic. Check AxoConsole for details.

A Flow used to have inbound or outbound traffic but currently has none. The alert fires per Flow and traffic direction. A Flow that has never received traffic doesn't trigger this alert; it only fires when previously flowing traffic has stopped. Common causes are an upstream source becoming silent, a configuration change that disabled or rerouted the source, or a Flow selector that no longer matches the intended sources.

Related metrics: logDiskQueueBytes, networkInputBytes

Source has no traffic (warning)

Source `hostname` used to send data regularly, but it has stopped. The Analytics view is a good starting point for investigation. To dismiss this alert, set the `traffic_alerts` label of the source to `false` in the inventory.

A source that used to send data regularly has stopped, for longer than the alert tolerates. A source that has never sent data doesn't trigger this alert. Common causes are a decommissioned or relocated device, a stopped log forwarder on the device, an address change that breaks host attribution, or a network path that no longer reaches the AxoRouter. Setting the `traffic_alerts` label of the source to `false` in the inventory clears the alert within a few minutes; removing the label enables it again. How much history the alert requires, how long a pause it tolerates, and whether it applies to a source carrying no label at all are deployment-wide settings.

Related metrics: logInputEvents, logInputBytes

AxoRouter service version mismatch (warning)

AxoRouter `hostname` has a service `service-id` (version `service-version`) that doesn't match its `axolet` 's version `version`. It may be due to a misconfigured deployment.

An AxoRouter host has a managed service whose binary version doesn't match the host's Axolet version. Service and Axolet versions normally move together; a mismatch usually points to a misconfigured deployment or to an in-progress upgrade that didn't complete. Contact Axoflow support if the mismatch doesn't clear on its own.

Suboptimal worker count (info)

Destination `id` on `hostname` is using more active partitions than workers. Consider increasing the number of workers.

An output destination on a host is using more active worker partitions than it has configured workers, which means it could be processing data with more parallelism than it has been allowed. The alert fires per destination. Increasing the worker count on the destination so it matches or exceeds the number of active partitions usually improves throughput.

Related metrics: logOutputEvents, eventDelaySeconds

Update available (info)

Axoflow software running on `hostname` has an update available, as it's running Axolet `version` while `version` is available.

A host is running a different Axolet version than the one the AxoConsole offers. An update is available and can be applied through the standard Axolet upgrade procedure. The AxoRouter service version normally moves together with the Axolet version, so this alert usually also implies that a new AxoRouter version is available (see also the `AxoRouterServiceVersionMismatch` alert). While the host runs a version that the AxoConsole no longer supports, the `UnsupportedVersion` alert replaces this one.

Unsupported version (warning)

An AxoRouter host runs an Axolet version that this AxoConsole no longer supports. The AxoConsole doesn't update its configuration until you upgrade it to the current version.

An AxoRouter host runs a version older than the oldest one the AxoConsole generates configuration for. The AxoConsole stops updating the configuration of its AxoRouter, Windows Event Collector and AxoStore services, so the host keeps its last configuration and does not pick up any change until it is upgraded through the standard Axolet upgrade procedure. This alert replaces `UpdateAvailable` for the host.

Host service has a missing file (warning)

Host `hostname` has a missing file which is required to write/reload config.

A managed service on a host can't have its configuration written or reloaded because a file it depends on is missing on the host. The alert fires per missing file, so a host with several missing files produces several alerts. The specific file name is available in the alert details. Place it in the expected location to resolve the issue.

Traffic without flows (warning)

There are no flows that have a router selector matching `hostname`, but it's receiving traffic. AxoRouter is dropping these events. Create a flow that matches it.

An AxoRouter is receiving input events, but no enabled Flow has a Router selector that matches it, so all received events are dropped. Either enable or create a Flow whose selector matches this AxoRouter, adjust the AxoRouter's labels so an existing Flow matches it, or create a Flow that explicitly drops these events if the traffic is unwanted.

Related metrics: droppedPacketsTotal, logInputBytes, logInputEvents, logOutputBytes, logOutputEvents, logMemoryQueue, logDiskQueue, logMemoryQueueBytes, logDiskQueueBytes, networkInputBytes, networkInputPackets, networkOutputBytes, networkOutputPackets

no `axorouter-syslog` service running (warning)

AxoRouter '`hostname`' has no `axorouter-syslog` running

An AxoRouter host has no running `axorouter-syslog` service registered with the AxoConsole. The AxoConsole can only configure an AxoRouter that has at least one such managed service present and running. This usually happens when the service fails to start on the host or when its registration is incomplete. It can also indicate that automatic service registration has been turned off.

AxoRouter Config Generation Error (warning)

AxoRouter configuration generation failed on '`hostname`' for service '`service-id`'. Check the `axorouter_confgen` controller logs in `controller-manager` for details.

Configuration generation for the AxoRouter service on a host has been failing for at least five minutes. The host continues to use the last successfully generated configuration, but changes made in the AxoConsole don't reach this host until generation succeeds again. This typically indicates an internal issue; the AxoConsole logs contain the underlying error, otherwise contact Axoflow support.

AxoStore Config Load Error (warning)

AxoStore configuration load failed on '`hostname`' for service '`service-id`'. Check the AxoStore logs for details.

The AxoStore service on a host failed to load its configuration. The previously loaded configuration remains in effect, so storage continues to operate, but configuration changes won't take effect on this host until the load succeeds. The AxoStore logs on the host contain the underlying reason.

WEC Unreachable Clients (info)

AxoRouter '`hostname`' has `value` unreachable clients on WEC subscription '`subscription`'. Reach out to Axoflow support on how to list unreachable clients.

An AxoRouter has one or more Windows Event Collector clients that have stopped sending heartbeats and are now considered unreachable. The alert fires per WEC subscription. Common causes are the WEC client service being stopped on the endpoint, a network problem between the endpoint and the AxoRouter, or a credential or policy change that broke authentication. Contact Axoflow support to obtain the list of unreachable clients.

Axolet frequently restarts (warning)

Axolet '`hostname`' has restarted >3 times in the last 15 minutes. Check the startup logs.

The Axolet process on a host has restarted more than three times in the last 15 minutes, which suggests a crash loop or repeated process termination. The host's startup logs typically contain the underlying reason; contact Axoflow support if the cause isn't obvious.

Host service frequently restarts (warning)

' `service-id` ' on ' `hostname` ' has restarted >3 times in the last 15 minutes. Check the startup logs.

A managed service on a host has restarted more than three times in the last 15 minutes, which suggests a crash loop or repeated process termination. The alert fires per service. The host's startup logs and the service's own logs typically contain the underlying reason; contact Axoflow support if the cause isn't obvious.

Rejected syslog connections (warning)

AxoRouter ' `hostname` ' has rejected connections in the last 5 minutes. Check the maximum connections in the connector rule settings for ' `id` '.

An AxoRouter has rejected incoming syslog source connections in the last five minutes because the configured maximum number of concurrent connections has been reached. The alert fires per syslog connector. If the increased connection count is legitimate, raise the **Maximum connections** setting on the syslog connector in the AxoConsole.

Related metrics: connections, logInputEvents

Syslog transport errors (warning)

An AxoRouter repeatedly rejected connections or messages on one of its syslog connectors in the last 5 minutes, always with the same error. Check the connector's settings against the configuration of the clients that send to it. For a TLS handshake error, see mutual authentication using TLS. For an invalid frame header, check the message framing, see the transport option.

Fires when one syslog connector reports the same transport error more than ten times in five minutes. The alert groups by connector and error kind, so each distinct error on a connector alerts separately.

Related metrics: connections, logInputEvents

Log source can't send logs (warning)

A log source sent no message to a syslog connector in the last 5 minutes, only transport errors, so none of its logs arrive. Check the source's client configuration against the connector's settings. For a TLS handshake error, see mutual authentication using TLS. For an invalid frame header, check the message framing, see the transport option.

Fires per source address and error kind, aggregated over AxoRouters, when a source causes at least three transport errors on one connector in five minutes and delivers no message on it in the same window. The AxoRouter name appears only when exactly one AxoRouter is affected.

Related metrics: connections, logInputEvents

Small UDP socket buffer (warning)

'`hostname`' has an UDP connector with a low socket buffer size configured.

An AxoRouter has at least one UDP connector with a socket receive buffer smaller than 8 MiB, and is actually receiving traffic on it. An undersized receive buffer is a common cause of UDP packet loss under load. Raise the `net.core.rmem_max` `sysctl` on the host operating system or increase the socket buffer size in the connector settings; the recommended size is 32 MiB.

Related metrics: droppedPacketsTotal, logInputEvents

Conflicting flow control parameters on a host (info)

Conflicting flow control parameters on `hostname`. The combination causes unintended backpressure and reduces throughput. This alert triggers when a router forwards traffic between a source and a destination with **Batch Timeout** > 0, disk buffering turned off, and **Log window size** / **Maximum connections** less than **Batch Lines** * **Number of Workers**. See the alert description for remediation steps.

An AxoRouter forwards traffic between a source and a destination whose flow-control parameters conflict: the destination has **Batch Timeout** > 0, disk buffering is disabled, and **Log window size** / **Maximum connections** is less than **Batch Lines** × **Number of Workers**. This combination can cause unintended backpressure and reduce throughput. To resolve, do one of the following:

- increase **Log window size** on the source, or
- reduce **Maximum connections** on the source, or
- reduce **Batch Lines** on the destination, or
- reduce **Number of Workers** on the destination.

Conflicting flow control parameters on a destination. (info)

Conflicting flow control parameters on a destination: `destination-name`. The combination causes unintended backpressure and reduces throughput. This alert triggers when a router forwards traffic between a source and a destination with **Batch Timeout** > 0, disk buffering turned off, and **Log window size** / **Maximum connections** less than **Batch Lines** * **Number of Workers**. **Batch Lines** or **Number of Workers** is too high to be continuously fed by messages from source connectors.

A destination receives traffic from a source whose flow-control parameters conflict with the destination's: the destination has **Batch Timeout** > 0, disk buffering is disabled, and **Log window size** / **Maximum connections** is less than **Batch Lines** × **Number of Workers**. **Batch Lines** or **Number of Workers** is too high to be continuously fed from the source connectors, which can cause unintended backpressure and reduce throughput. To resolve, do one of the following:

- increase **Log window size** on the source, or
- reduce **Maximum connections** on the source, or
- reduce **Batch Lines** on the destination, or
- reduce **Number of Workers** on the destination.

Low disk free space (warning)

There is low disk free space on `hostname`.

A disk used by the AxoStore on a host has less than 10% free space. The alert fires per disk, so a host with multiple low disks produces multiple alerts. If a disk fills up entirely, the AxoStore on this host will be unable to persist new messages.

Store's disk is filling up (warning)

Store's disk is filling up on `hostname`

A disk used by the AxoStore on a host is filling up fast enough that, at the current rate, it's predicted to be full within the next six hours. The alert fires per disk. Once a disk fills up entirely, the AxoStore on this host will be unable to persist new messages.

Store's error rate increased (warning)

Store's error rate increased on `hostname`

The rate at which the AxoStore on a host emits errors of a given type has risen above the expected baseline. The alert fires per error type, so a single underlying problem may surface as several concurrent alerts. A persistent high error rate often degrades storage reliability and should be investigated promptly.

AxoStore doesn't persist messages (critical)

AxoStore doesn't persist messages from `axorouter-syslog` .

An AxoRouter is receiving traffic that's configured to be persisted in the AxoStore, but the AxoStore was unable to persist it. The unpersisted messages aren't available in the AxoConsole for tapping or search. Check the AxoConsole for the underlying reason and remediation steps.

11.6 Private connections

Connect your cloud hosts to AxoConsole over private links, like Google Private Service Connect, without using the public internet.

11.6.1 Google Private Service Connect

Connect Google Cloud hosts to AxoConsole through Private Service Connect, so their traffic stays on the Google network.

If you want your hosts in Google Cloud to access the AxoConsole without leaving the Google network, we recommend that you use Google Cloud Private Service Connect (PSC) to secure the connection from your VPC to Axoflow.

Prerequisites

Contact Axoflow and provide the list of projects so we can set up an endpoint for your PSC. You will receive information from us that you'll need to properly configure your connection.

You will also need to allocate a dedicated IP address for the connection in a subnet that's accessible for the hosts.

Steps

After you have received the details of your target endpoint from Axoflow, complete the following steps to configure Google Cloud Private Service Connect from your VPC to AxoConsole.

1. Open Google Cloud Console and navigate to Private Service Connect > Connected endpoints.
2. Select the project you want to connect to Axoflow.
3. Navigate to Connect endpoint and complete the following steps.
    1. Select **Target > Published service**.
    2. Set **Target service** to the service name you've received from Axoflow. The service name should be similar to: `projects/axoflow-shared/regions/<region-code>/serviceAttachments/<your-tenant-ID>`
    3. **Set Endpoint name** to the name you prefer, or the one recommended by Axoflow. The recommended service name is similar to: `psc-axoflow-<your-tenant-ID>`

4. Select your VPC in the **Network** field.
5. Set **Subnet** where the endpoint should appear. Since subnets are regional resources, select a subnet in the region you received from Axoflow.
6. Select **Create IP address** and allocate an address for the endpoint. Save the address, you'll need it later to verify that the connection is working.
7. Select **Enable global access**.
8. There is no need to enable the **directory API** even if it's offered by Google.
9. Select **Add endpoint**.

4. Test the connection.

    1. Log in to a machine where you want to use the PSC using SSH.
    2. Test the connection. Run the following command using the IP address you've allocated for the endpoint.

        ```
        curl -vk https://<IP-address-allocated-for-the-endpoint>
        ```

        If the connection is established, you'll receive an HTTP 404 response.

5. If the connection is established, configure DNS resolution on the hosts either for selected machines or the whole VPC.

Setting up selected machines to use the PSC

1. Add the following entry to the `/etc/hosts` file of the machine.

    ```
    <IP-address-allocated-for-the-endpoint> <your-tenant-id>.cloud.axoflow.io kcp.
    <your-tenant-id>.cloud.axoflow.io telemetry.<your-tenant-id>.cloud.axoflow.io
    ```

2. Run the following command to test DNS resolution:

    ```
    curl -v https://<your-tenant-id>.cloud.axoflow.io
    ```

    It should load an HTML page from the IP address of the endpoint.

3. If the host is running axolet, restart it by running:

```
sudo systemctl restart axolet.service
```

Check the axolet logs to verify that there're no errors:

```
sudo journalctl -fu axolet
```

4. Deploy the changes of the `/etc/hosts` file to all your VMs.

Setting up whole VPC networks to use the PSC

1. Open Google Cloud Console and in the *Cloud DNS* service navigate to the *Create a DNS zone* page.
2. Create a new *private* zone with the *zone name* `<your-tenant-id>.cloud.axoflow.io`, and select the networks you want to use the PSC in.
3. Add the following three *A records*, all of which targeted to the `<IP-address-allocated-for-the-endpoint>`:
   - `<your-tenant-id>.cloud.axoflow.io`
   - `kcp.<your-tenant-id>.cloud.axoflow.io`
   - `telemetry.<your-tenant-id>.cloud.axoflow.io`

12 Metrics and analytics

Access the metrics and analytics that Axoflow collects about your hosts, flows, storage, and usage across your security data pipeline.

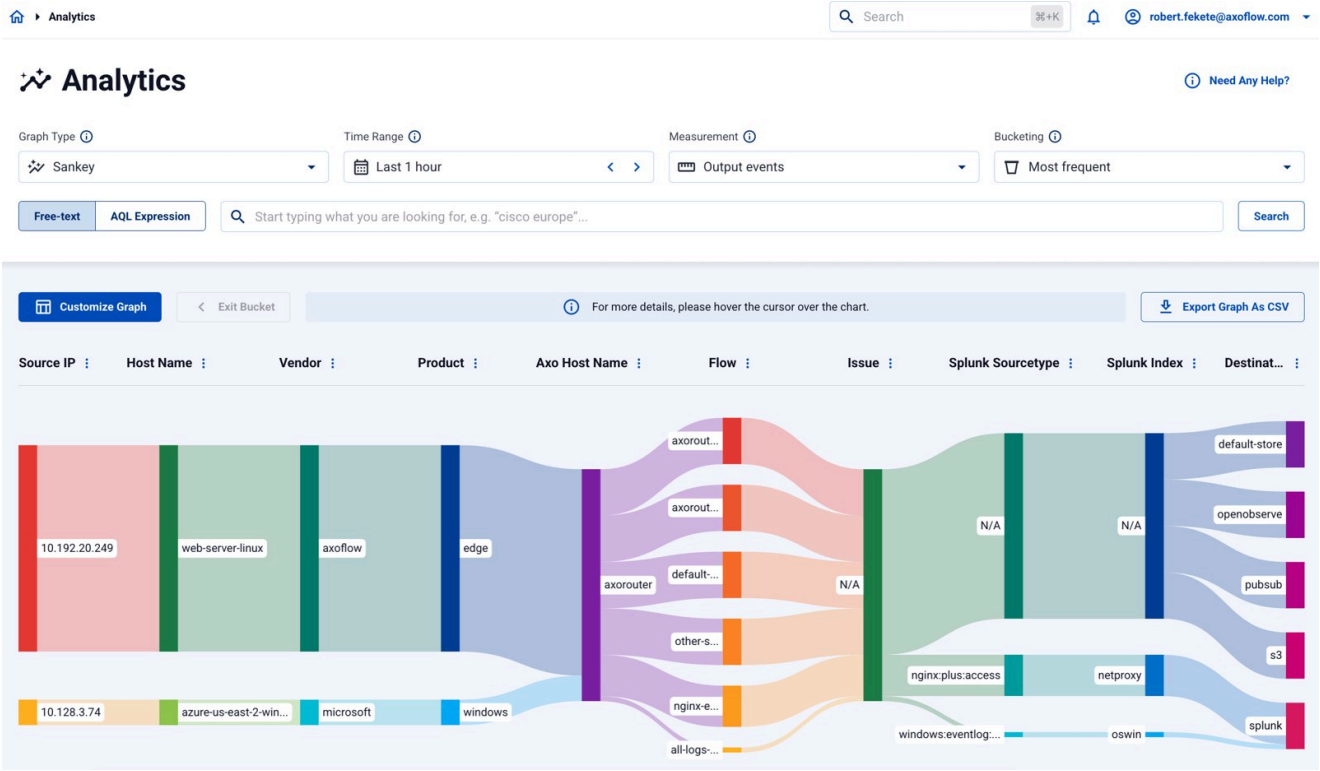
This chapter shows how to access the different metrics and analytics that Axoflow collects about your security data pipeline.

12.1 Analytics

Analyze the data of your pipeline with Sankey, Sunburst, and time series charts, filter it by labels, and tap into the log flow.

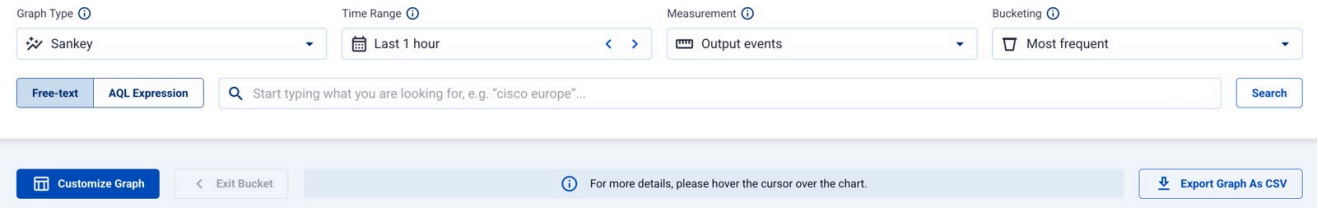
Use Sankey, Sunburst, and Time series diagrams to analyze your data at various points in your Axoflow pipeline.

- Use the **Analytics** page to analyze the data throughput of your pipeline.
- You can analyze the throughput of a flow on the **Flows > <flow-to-analyze> > Analytics** page.
- You can analyze the throughput of a specific source on the **Sources > <source-to-analyze> > Analytics** page.
- You can analyze the throughput of a specific AxoRouter on the **Routers > <router-to-analyze> > Analytics** page.



Configure the analytics charts

Use the top bar to select the dataset that you want to work with. Use **Customize Graph** to select the labels to show for the selected dataset.



- **Time Range**: Select the calendar_month icon to change the time period that's displayed on the charts. You can use absolute (calendar) time, or relative time (for example, the last 2 days). To move to the previous or the next period of the same length, select the arrows next to the time period (**Previous interval** and **Next interval**).

  Axoflow stores all dates in Coordinated Universal Time (UTC), and automatically converts it to the timezone of set in your browser/operating system.

- **Graph Type**: Switch between Sankey, Sunburst, and Time series diagrams.
- **Measurement**: Change the display of data throughput to:
  - **Output bytes**
  - **Output events**
  - **Input bytes**
  - **Input events**
- **Bucketing** (Sankey and Sunburst diagrams only): **Alphabetic**, **Most frequent** (top talkers first), or **Least frequent** (least significant contributors first).
- The settings of the filter bar change the URL parameters of the page, so you can bookmark it, or share a specific view by sharing the URL.

Select a segment of the diagram to drill down into the data. To go back, select **Exit Bucket**.

Hover over a segment to display its details.

Select **Export Graph As CSV** to download the currently visible dataset as a CSV file to process it in an external tool.

Search the analytics data

- **Free-text** mode searches in the values of the labels that are selected in **Customize Graph**, and filters the dataset to the matching data. For example, if the chart shows the **Host Name** and **Product** labels, and you enter a search keyword in **Free-text** mode, the chart shows only the data where the keyword appears in the hostname or the product name.

The search is case insensitive. Adding multiple keywords searches for matches in any of the previous fields. This is equivalent to the `@ANY =* keyword1 AND @ANY =* keyword2` AQL query.

- Use **AQL Expression** mode to filter the data by any available label.

  It also makes more complex filtering possible, using the Equals, Contains (partial match), and Match (regular expression match) operators. Note that:

  - To execute the search, click **Search**, or hit ESC then ENTER.
  - AxoConsole autocompletes the built-in and custom labels and field names, as well as their most frequent values, but doesn't autocomplete labels and variables created by data parsing and processing steps.
  - You can use the AND and OR operators to combine expressions, and also parenthesis if needed. For details on AQL, see AQL operator reference.
  - The precedence of the operators is the following: parentheses, AND, OR, comparison operators.
  - Use the usual keyboard shortcuts to undo (⌘/Ctrl + Z) or redo (⌘/Ctrl + Shift + Z) your edits.

Select the labels of analytics charts

Select **Customize Graph** to set which labels the chart shows. You can:

- Reorder the labels to adjust the diagram. On Sunburst diagrams, the left-most label is on the inside of the diagram.
- Add new labels to get more details about the data flow.

  Some labels, like **Flow** and **Destination Name**, are available only for **Output bytes** and **Output events**.

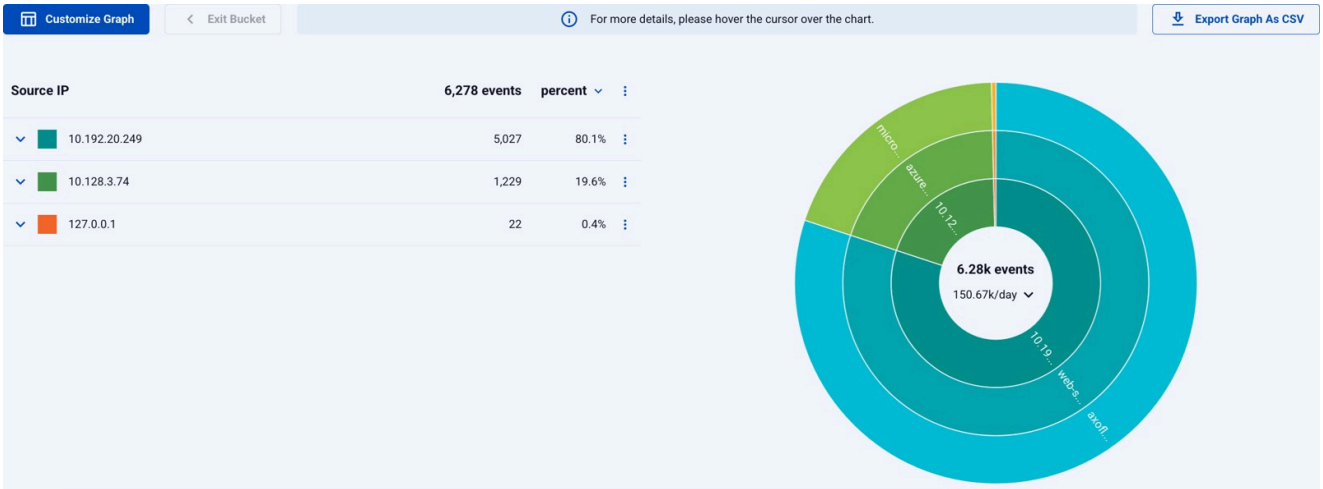
  - Labels added to AxoRouter hosts get the `axo_host_` prefix.
  - Labels added to data sources get the `host_` prefix. For example, if you add a *rack* label to an edge host, it'll be added to the data received from the host as `host_rack`.
  - Labels added on edge hosts get the `edge_connector_label_` prefix.

  On other pages, like the **Overview** tab of a host, the labels are displayed without the prefixes.
- Remove unneeded labels from the diagram.

For a list of the available labels, see Metrics schema.

Sunburst diagrams

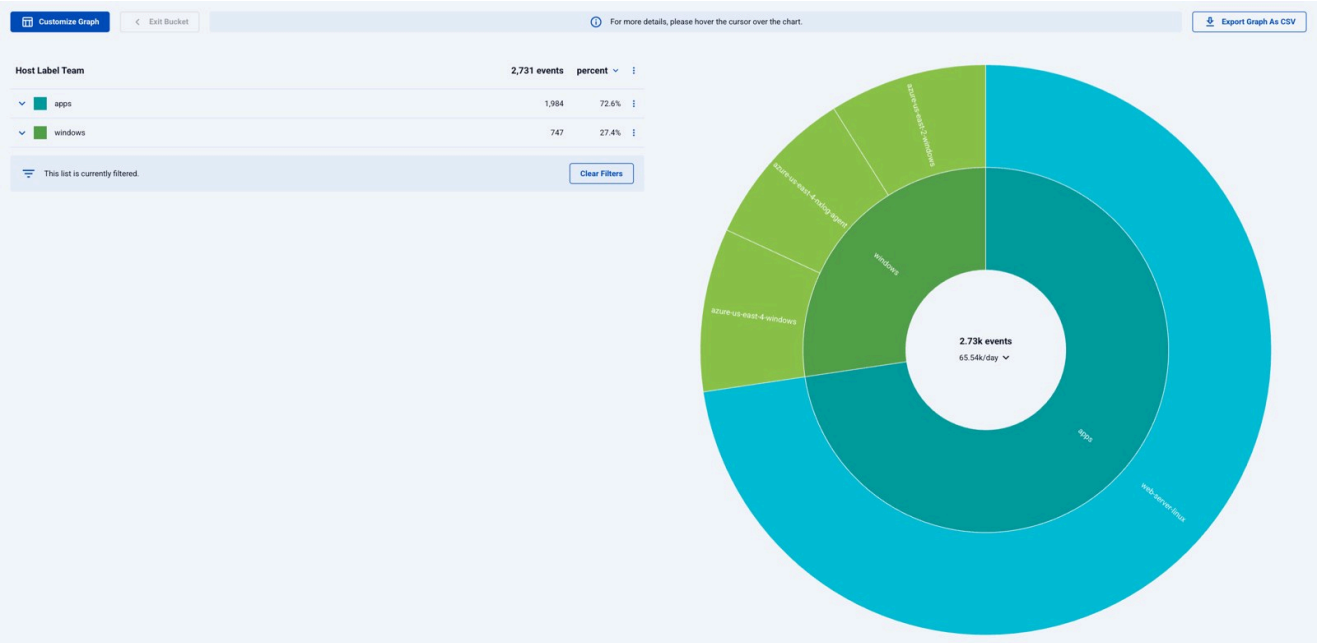
Sunburst diagrams (also known as ring charts or radial treemaps) visualize your data pipeline as a hierarchical dataset. They organize the data according to the labels enabled using **Customize Graph** into concentric rings, where each ring corresponds to a level in the hierarchy. The left-most label is on the inside of the diagram.



Use Sunburst diagrams to find, for example:

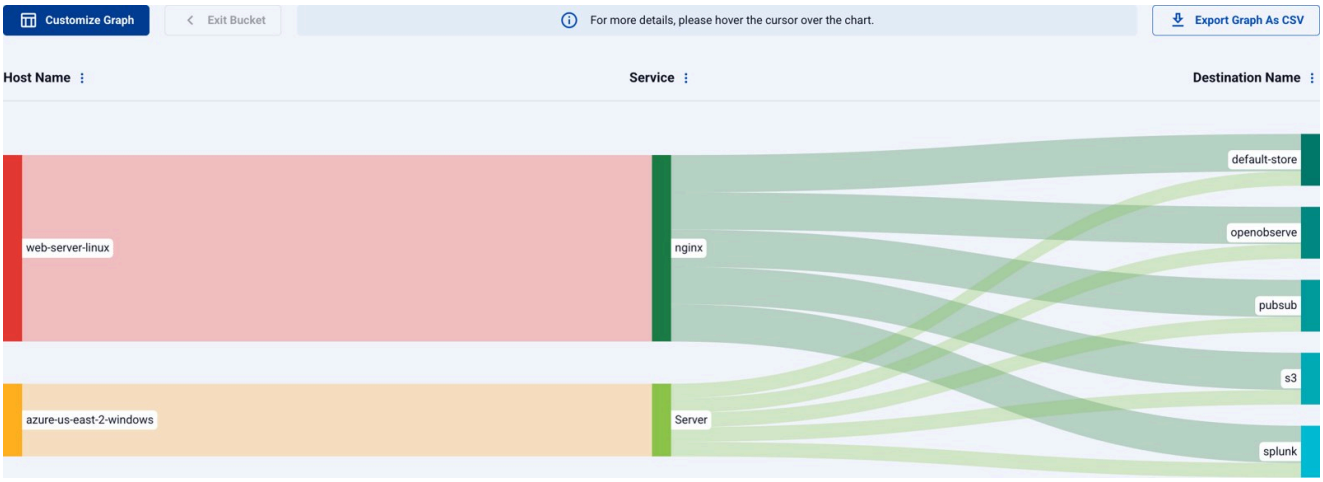
- the top talkers (the data sources that send the most data),
- the team that sends the most data to a destination, if you've added custom labels that show the owner of your data sources.

The following example groups the data sources that send data into a Splunk destination (filtered with the AQL expression `destination_name = "splunk"`) based on their custom `host_label_team` labels.



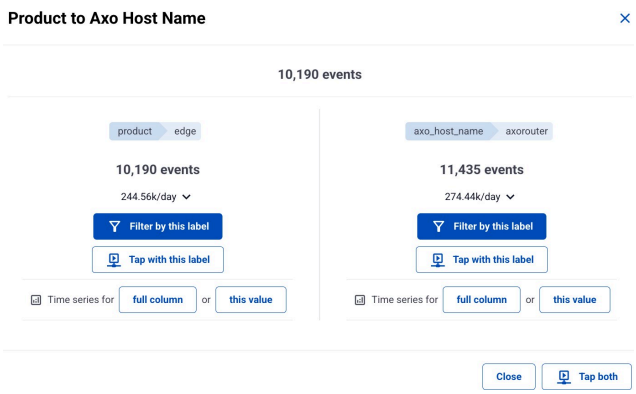
Sankey diagrams

The Sankey diagram of your data pipeline shows the flow of data between the elements of the pipeline, for example, from the source (host) to the destination. Sankey diagrams show how the data flow divides at each stage. That way, they help you find bottlenecks, and show where and how much data is flowing.

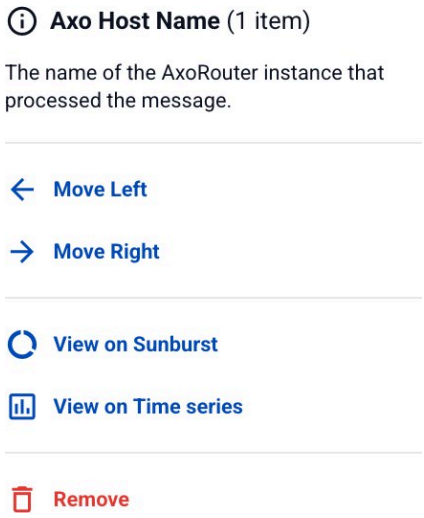


The diagram consists of nodes (also called segments) that represent the different attributes or labels of the data flowing through your pipeline. Nodes are shown as labeled columns, for example, **Host Name**, **Service**, or **Destination Name**. The thickness of the links between the nodes of the diagram shows the amount of data.

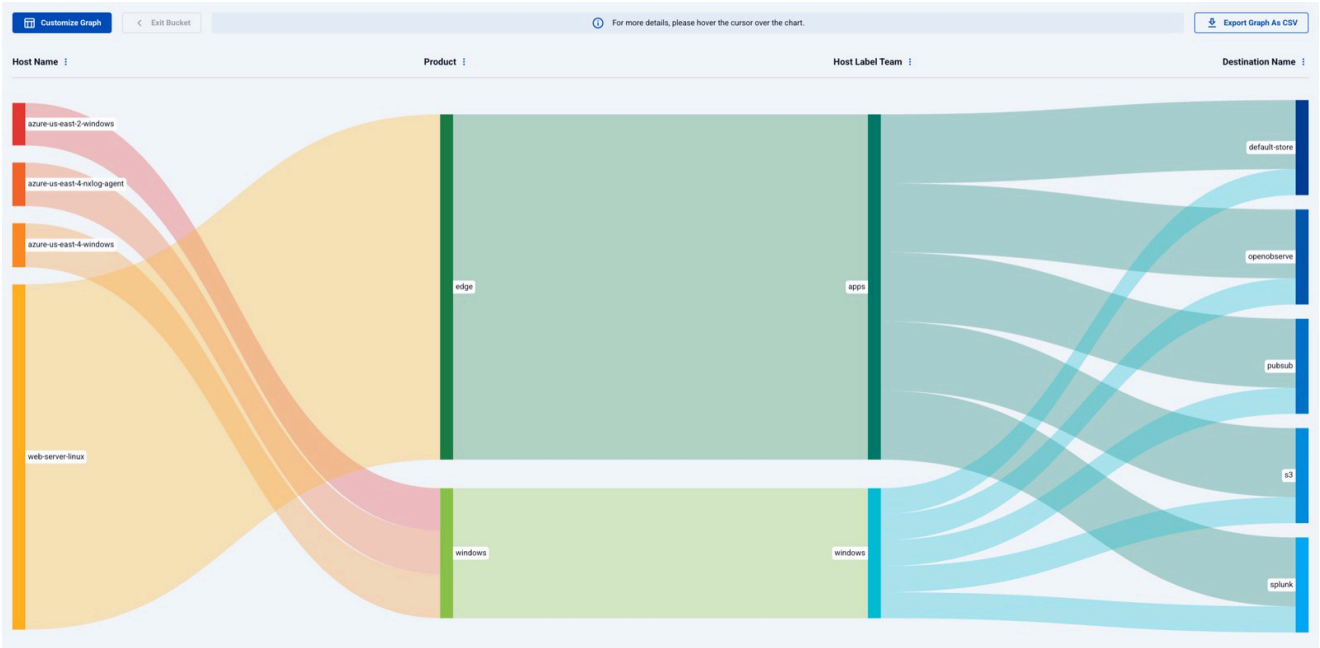
- Hover over a link to show the data throughput of this link between the nodes of the diagram.
- Select a link to:
  - Show the details of the link: the labels that the link connects, and their data throughput.
  - Filter the diagram to one of the labels: select **Filter by this label**.
  - Tap into the log flow.
  - Open a Time series chart.



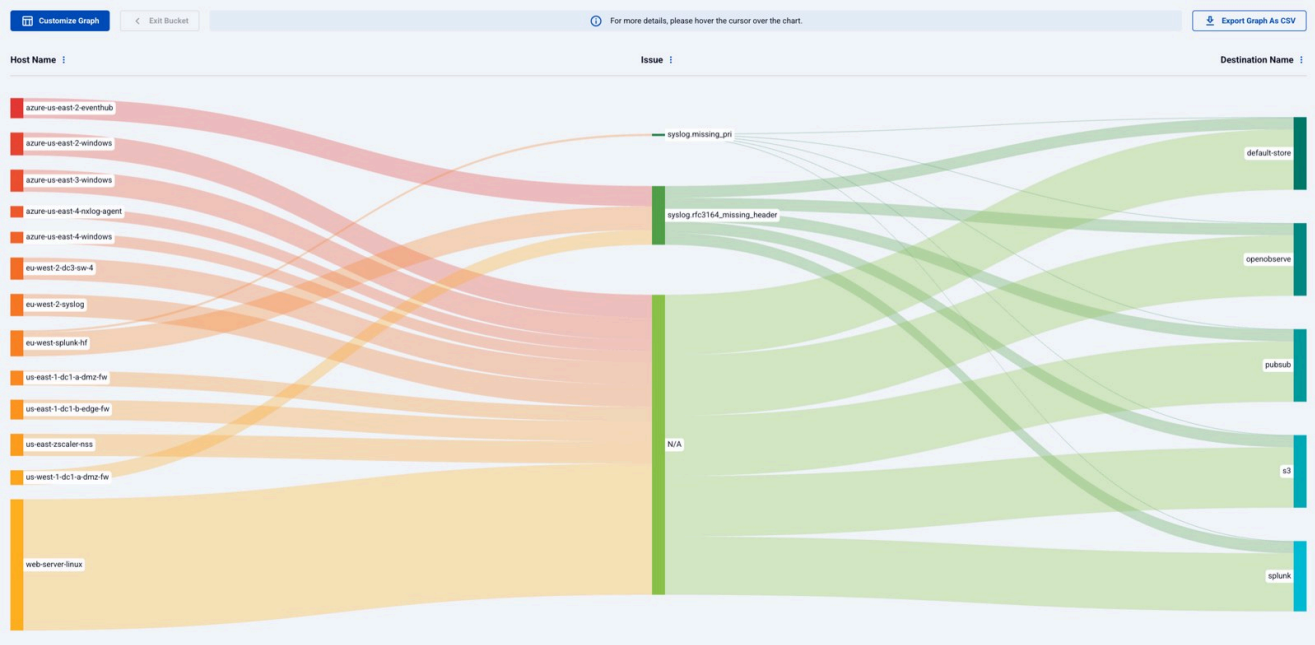
- Select a node to drill down into the diagram. To undo, use the Back button of your browser, or select **Exit Bucket**.
- Select ⋮ next to a column name to open its menu. You can move the column with **Move Left** and **Move Right**, open the label in **View on Sunburst** or **View on Time series**, or **Remove** the label.



The following example shows a custom label that shows the owner of the source host, so you can see which team sends the most data to the destination.



The following Sankey diagram shows the `issue` label over the last 7 days. The label marks messages that were invalid or malformed for some reason (for example, the header of a syslog message was missing).

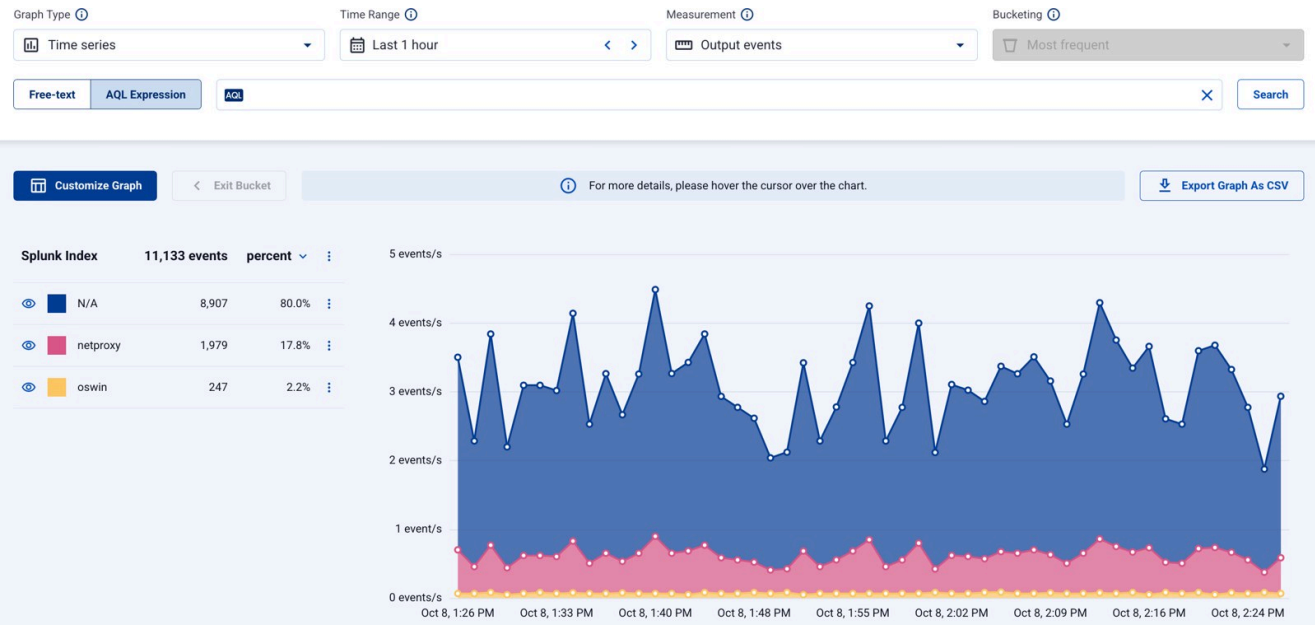


Use Sankey diagrams to:

- Visualize flows: add the **Flow** label using **Customize Graph**.
- Find unclassified messages that the Axoflow database didn't recognize: add the **Auto Classified Service** label using **Customize Graph**, and look for the `N/A` link. You can tap into the log flow to check these messages. Send sample messages to Axoflow support so that we can add them to the classification database.
- Visualize custom labels and their relation to data flows.

Time series charts

The Time series chart shows how the metrics of the selected labels change over time.

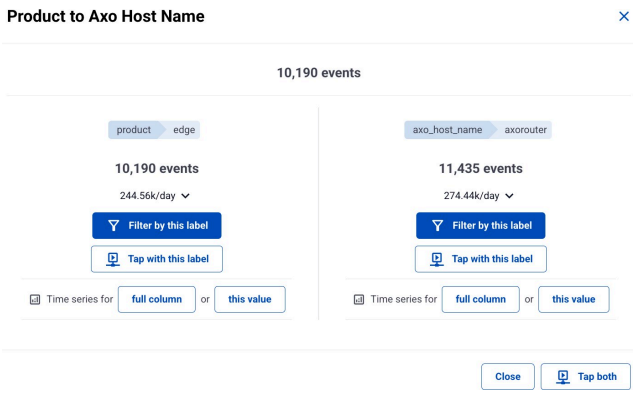


- Hover over the chart to show the metrics at that time.
- **Export Graph As CSV** downloads the currently visible dataset as time-series data.

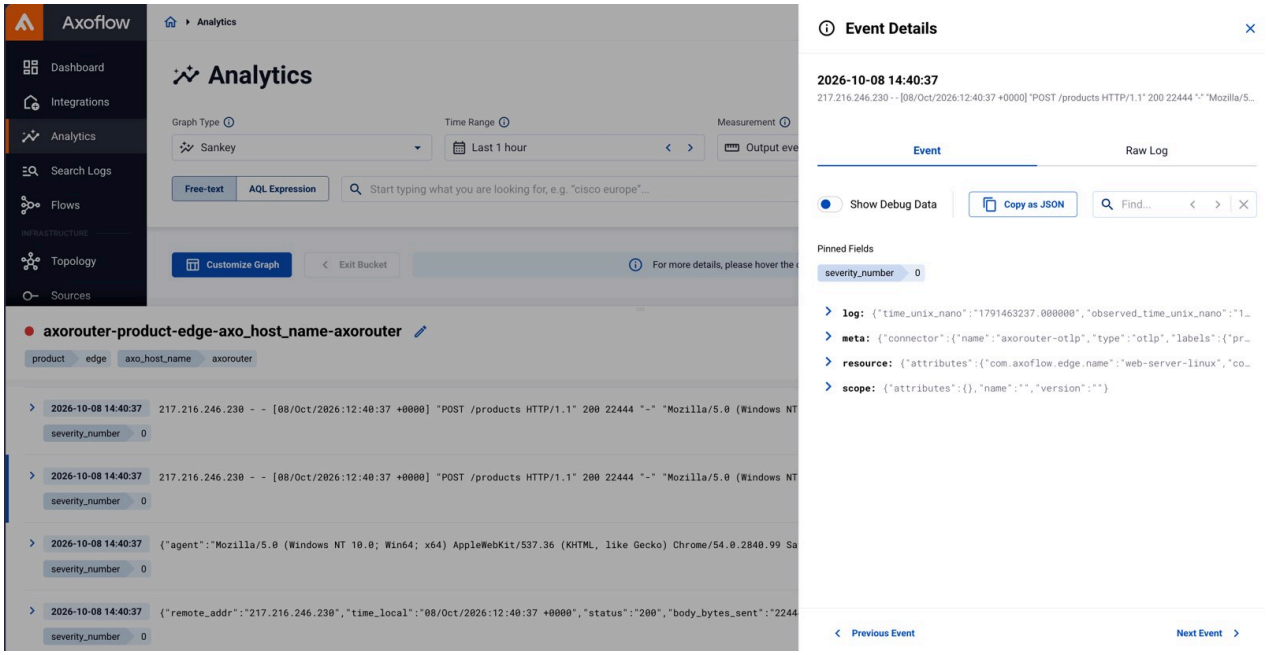
You can also open time series charts from Sankey diagrams: select a link in the diagram to open its details dialog. In the **Time series for** row, **full column** shows the chart for the entire Sankey column, and **this value** shows the chart for the selected link.

Tap into the log flow

1. On the Sankey diagram, select a link to show the details of the link.



2. Tap into the data traffic:
    - Select **Tap with this label** to tap into the log flow at either end of the link.
    - Select **Tap both** to tap into the data flowing through the link.
3. Select the host where you want to tap into the logs.
4. Select **Start**.
5. When the logs you're interested in show up, select **Pause**, then select a log message to see its details.

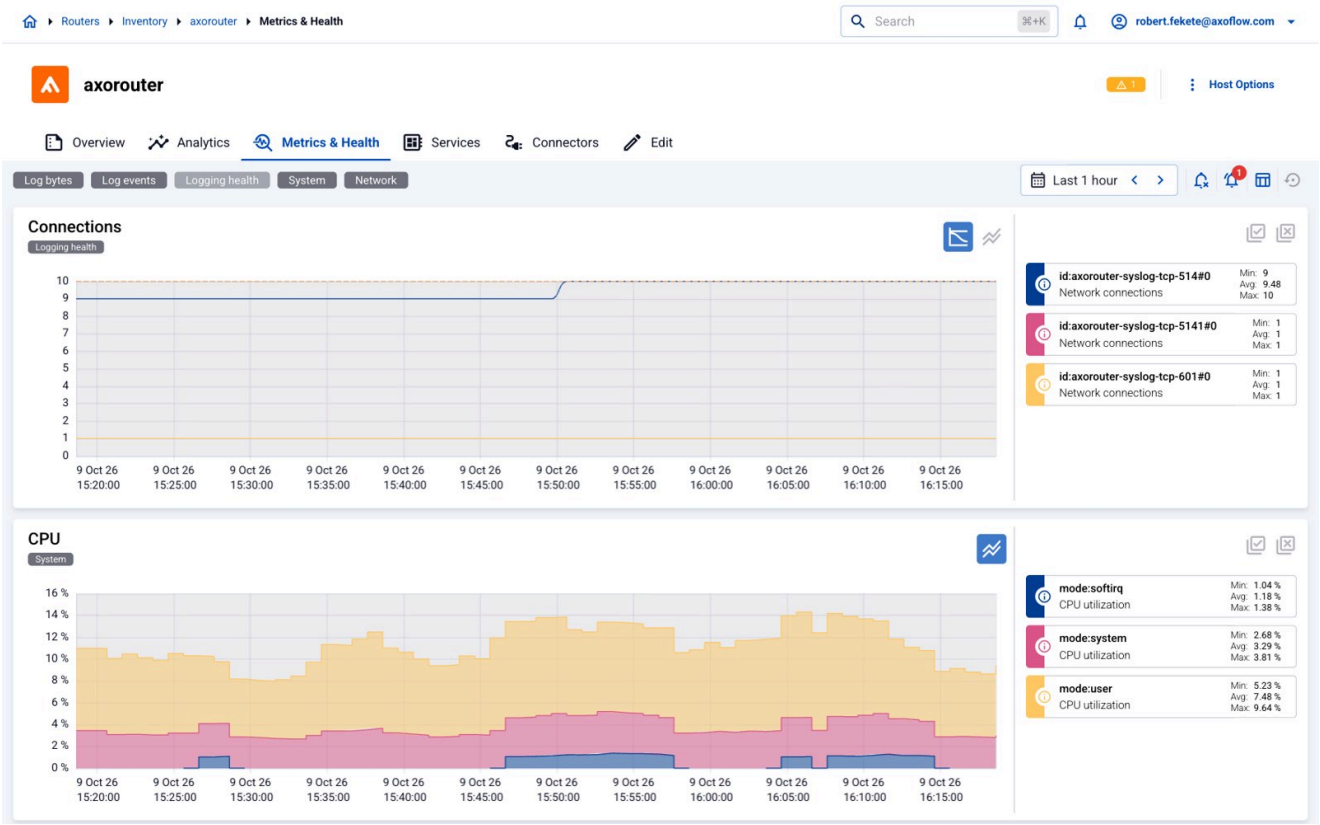


For details, see Log tapping.

12.2 Host metrics

Check the CPU, disk, connection, input, output, and queue metrics of a host over time, and see alerts and events on the charts.

The **Metrics & Health** page of a source or an AxoRouter host shows the history of the various metrics Axoflow collects about the host.



The charts also show the events of the host, for example, configuration reloads, or alerts that affect the host.



Filter and arrange host metrics

Use the bar above the charts to select which metrics to show, and for which time period.



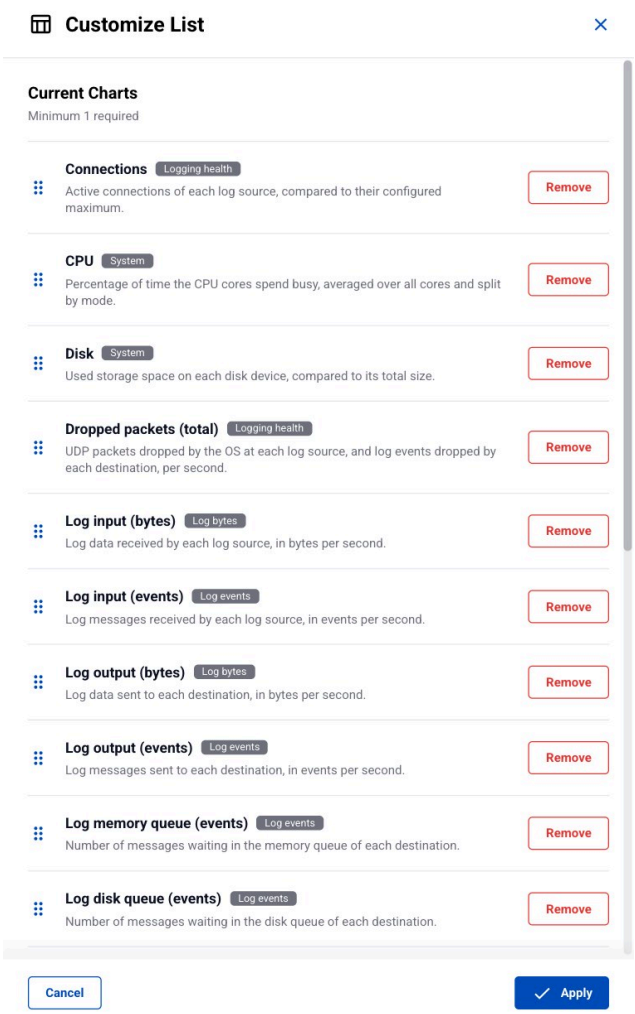
- *Metrics categories*: Hide or show the metrics of a category, for example, **System**. The category of each metric is under the name of its chart. This change isn't saved: to change the layout of the metrics, use table_chart **Customize List**.
- calendar_month **Time range**: Change the time period of the charts. You can use absolute (calendar) time, or relative time (for example, the last 2 days). To move to the previous or the next period of the same length, select the arrows next to the time period (**Previous interval** and **Next interval**).

  Axoflow stores all dates in Coordinated Universal Time (UTC), and automatically converts it to the timezone of set in your browser/operating system.

  > **Note**
  >
  > **By default, Axoflow stores metrics for 30 days, unless specified otherwise in your support contract. Contact us if you want to increase the data retention time.**

- **Hide all alerts** / **Show all alerts**: Hide or show the alerts and other events (like configuration reloads) of the host. By default, the charts show these events.
- notifications_active **Alerts**: Shows the number of active alerts on the host, and opens the list of alerts.
- table_chart **Customize List**: Change which charts are displayed, and in which order:
  - To change the order of the charts, drag a chart in the **Current Charts** list.
  - To hide a chart, select **Remove**.
  - To display a hidden chart, find it in the **Available Charts** list, and select **Add**.

  Then select **Apply**. These changes are persistent and stored in your profile.



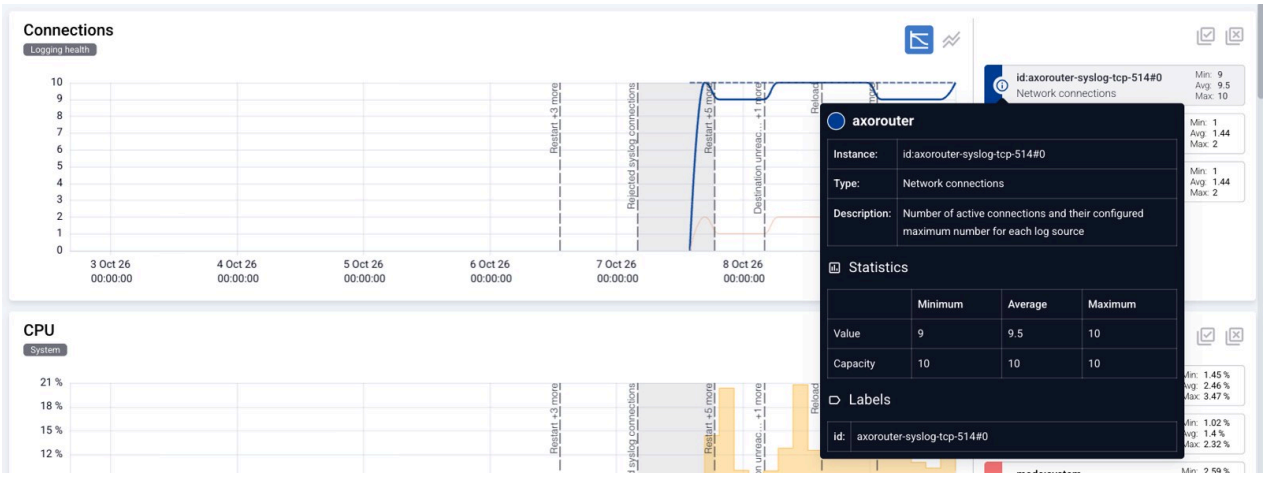
- settings_backup_restore **Restore default chart order**: Show the charts that you saved with **Customize List**, in the saved order. Use it to undo temporary changes, for example, the categories that you hid. The icon is active only when the displayed charts differ from the saved list.

The settings of the filter bar change the URL parameters of the page, so you can bookmark it, or share a specific view by sharing the URL.

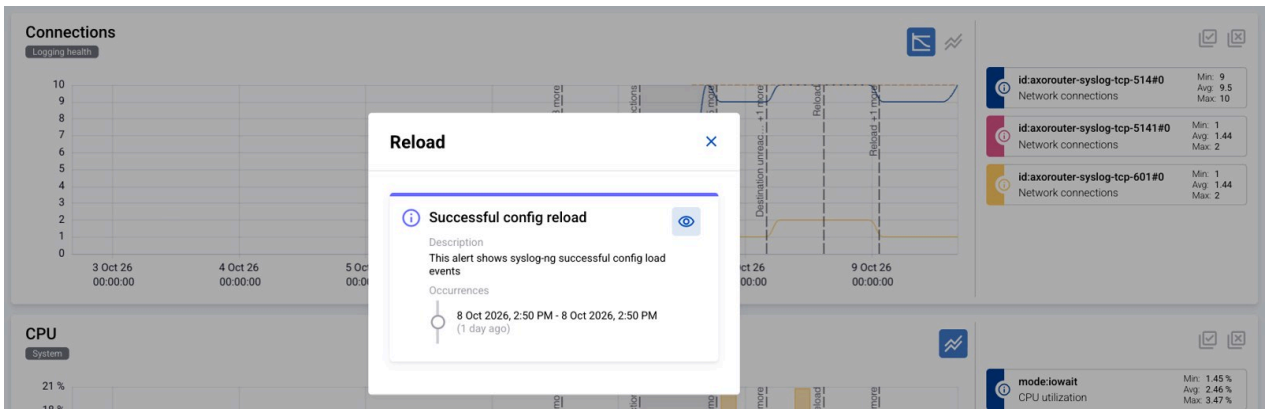
Interact with a host metrics chart

In addition to the controls of the top bar, you can interact with the charts in these ways:

- Hover over the info icon on a colored metric card to display the definition, details, and statistics of the metric.



- Select a colored card to hide the related metric from the chart. For example, you can hide unneeded sources on the **Log input (bytes)** and **Log input (events)** charts.
- Select library_add_check **Show all** or **Hide all** above the cards to show or hide every metric of the chart.
- Select stacked_line_chart **Show stacked graph** / **Show unstacked line graph** to switch between a stacked graph and an unstacked line graph.
- On charts that show a capacity (for example, **Connections** and **Disk**), select the capacity icon (**Show capacity** or **Hide capacity**) to show or hide the capacity line.
- Select an event (for example, an alert or configuration reload) to show its details.



- To quickly zoom in on a period, click and drag to select the period to display on any of the charts. Every chart will be updated for the selected period. (To return to the previous state, click the Back button of your browser.)

Host metrics reference

For managed hosts, the following metrics are available, in their default order on the page:

- **Connections**: Active connections of each log source, compared to their configured maximum.
- **CPU**: Percentage of time the CPU cores spend busy, averaged over all cores and split by mode, within a time window of 5 minutes.
- **Disk**: Used storage space on each disk device, compared to its total size.
- **Dropped packets (total)**: UDP packets dropped by the OS at each log source, and log events dropped by each destination, per second. The chart shows these metrics:
  - **Dropped UDP packets**: Number of UDP packets that the OS dropped before processing, per second, averaged within a time window of 5 minutes.
  - **Dropped log events**: Number of events dropped from event queues within a time window of 5 minutes.
- **Log input (bytes)**: Log data received by each log source, in bytes per second, averaged within a time window of 5 minutes.
- **Log input (events)**: Log messages received by each log source, in events per second, averaged within a time window of 5 minutes.
- **Log output (bytes)**: Log data sent to each destination, in bytes per second, averaged within a time window of 5 minutes.
- **Log output (events)**: Log messages sent to each destination, in events per second, averaged within a time window of 5 minutes.
- **Log memory queue (events)**: Number of messages waiting in the memory queue of each destination.
- **Log disk queue (events)**: Number of messages waiting in the disk queue of each destination.
- **Log memory queue (bytes)**: Memory used by messages waiting in the memory queue of each destination.
- **Log disk queue (bytes)**: Disk space used by the disk queue of each destination, compared to its capacity, plus the memory caching it. The chart shows these metrics:
  - **Disk queue bytes**: Total bytes of data waiting in each disk queue, and their configured capacity.
  - **Disk queue memory cache bytes**: Amount of memory used for caching disk-based queues.
- **Memory**: Memory in use, compared to the total memory reported by the OS (including reclaimable caches and buffers).

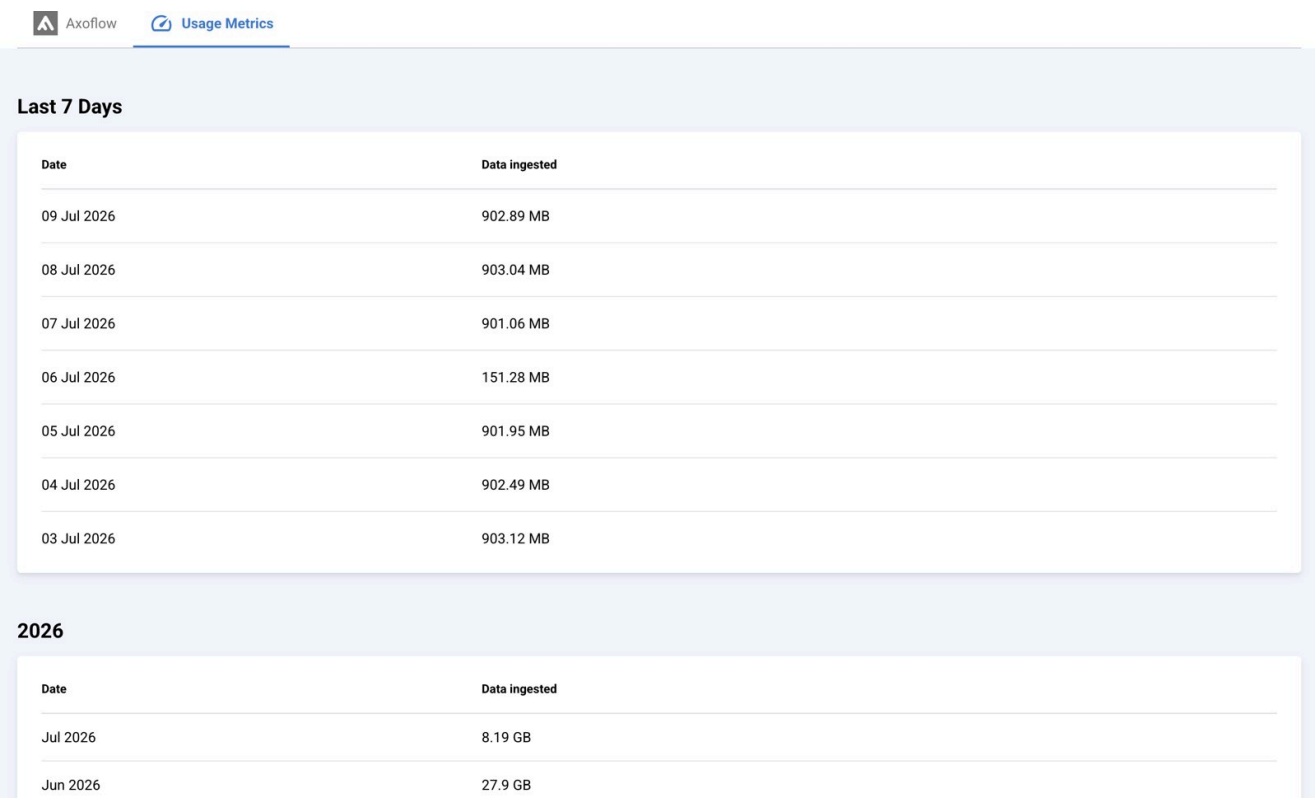
- **Network input (bytes)**: Incoming network traffic on each network interface, in bytes per second, averaged within a time window of 5 minutes.
- **Network input (packets)**: Incoming network packets on each network interface, per second, averaged within a time window of 5 minutes.
- **Network output (bytes)**: Outgoing network traffic on each network interface, in bytes per second, averaged within a time window of 5 minutes.
- **Network output (packets)**: Outgoing network packets on each network interface, per second, averaged within a time window of 5 minutes.
- **Event delay (seconds)**: Average time from receiving a message until it's sent to each destination. The chart is hidden by default. To show it, add it in **Customize List**.

12.3 Usage metrics

Check the daily and monthly data volume that your AxoRouter instances ingest, which Axoflow uses for billing.

The **About > Usage Metrics** page shows how much data your AxoRouter instances ingest. Axoflow uses these numbers for billing.

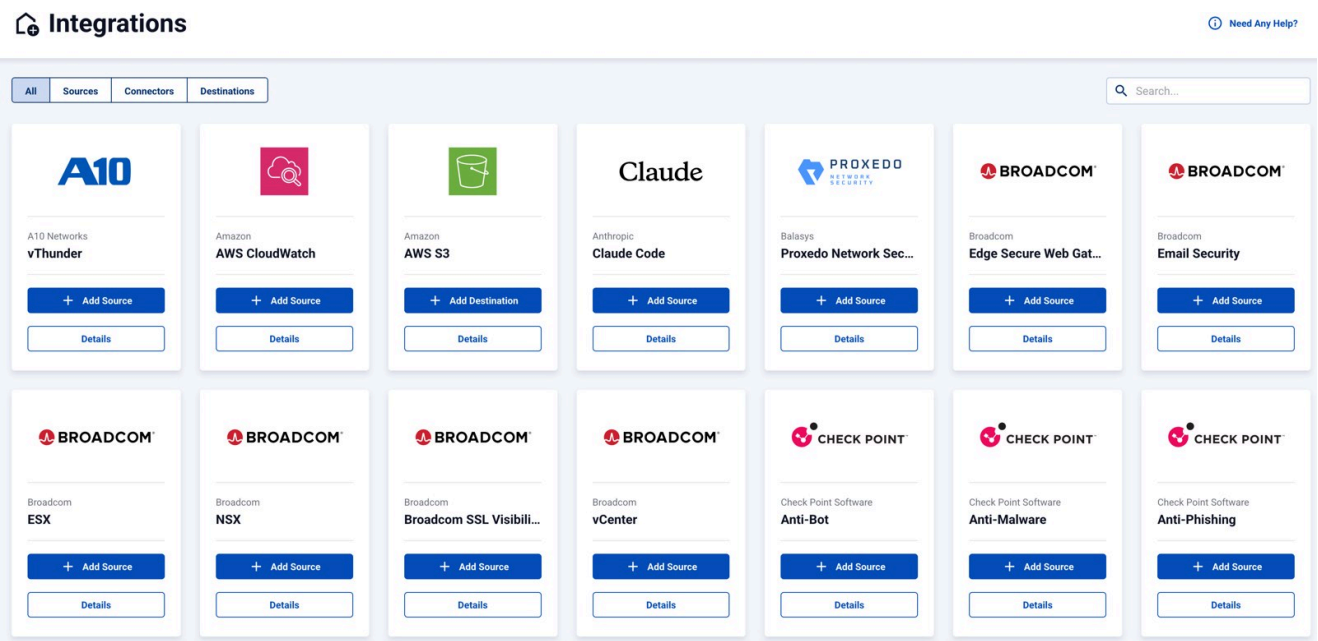
- **Last 7 Days**: the data ingested on each day.
- **<year>** (for example, **2026**): the data ingested in each month of the year.



13 Integrations

The source vendors, products, and destinations that Axoflow supports.

The **Integrations** page of AxoConsole lists all the sources, connector rules, collection rules, and destinations that Axoflow supports. You can configure each integration directly from the list.



Each card shows the vendor, the product, and its role:

- **Source**: A data source that sends data to Axoflow.
- **Connector Rule**: A connector rule that configures AxoRouter to receive data.
- **Collection Rule**: A collection rule that configures AxoEdge to collect data.
- **Destination**: A destination where Axoflow sends data.

The **Kubernetes**, **Linux**, and **Windows** cards of the Edge vendor open the provisioning page, where you can deploy an Axoflow agent agent.

A product that supports several roles appears once for each role.

To use the page:

- To filter the list by role, select **All**, **Sources**, **Connectors**, or **Destinations**. The **Connectors** filter shows both connector rules and collection rules.
- To find an integration, type the name of the product or the vendor into the **Search** field.
- To see the description of an integration and a link to its documentation, select **Details**.

- To configure an integration, select **Add Source**, **Add Connector Rule**, **Add Collection Rule**, or **Add Destination** on its card.

Supported products

The following list shows the source vendors, products, and destinations that Axoflow supports.

1Password

Source

1Password

Manages and secures user credentials, secrets, and vaults for individuals and organizations.

Contact support

A10

Source

A10 Networks vThunder

Delivers application load balancing, traffic management, and DDoS protection for enterprise networks.

Documentation

Amazon Web Services

Contact support



Source

Amazon AWS CloudTrail

Tracks AWS account activity and API usage for auditing, governance, and compliance monitoring.

Documentation



Source

Amazon AWS CloudWatch

Monitors AWS resources and applications by collecting metrics, logs, and setting alarms.

Documentation



Destination

Amazon AWS S3

Scalable cloud storage service for storing and retrieving any amount of data via object storage.

Documentation



Destination

Amazon AWS Security Lake

Centralizes and normalizes security data from AWS and third-party sources for analytics and compliance.

Contact support



Source

Amazon AWS VPC Flow Logs

Records the IP traffic going to and from network interfaces in a VPC.

Documentation

Claude

Source

Anthropic Claude Code

Agentic coding tool emitting OpenTelemetry audit events for tool use, permissions, hooks, and MCP connections.

Documentation

THE ASF

Apache Software Foundation Velocity

Contact support

macOS

Apple macOS

Contact support

Bitbucket

Atlassian Bitbucket

Contact support

AXOFLOW LOGGING UNLEASHED

Axoflow AxoRouter

Manages and routes event flow through pipelines, including filtering, parsing, and transformations.

Documentation

AXOFLOW LOGGING UNLEASHED

Axoflow AxoStore

AxoStore is a lightweight, queryable, temporary storage option that stores data locally on the AxoRouter host. AxoConsole provides federated search for AxoStore.

Documentation

AXOSYSLOG

Axoflow AxoSyslog

High-performance, configurable syslog service for collecting, processing, and forwarding log data.

Documentation

AXOFLOW LOGGING UNLEASHED

Axoflow Telemetry Controller

Telemetry Controller collects, routes and forwards telemetry data (logs, metrics and traces) from Kubernetes clusters supporting multi-tenancy out of the box.

Contact support

PROXEDO NETWORK SECURITY

Source

Balasys Proxedo Network Security

Application-level proxy firewall suite that granularly inspects and controls network traffic at the protocol level.

Documentation

box

Source

Box, Inc. Box

Cloud-based content management and file sharing platform designed for secure collaboration and storage.

Contact support

BROADCOM

Source

Broadcom Edge Secure Web Gateway (Edge SWG)

Secures web traffic through policy enforcement, SSL inspection, and real-time threat protection.

Documentation

BROADCOM

Source

Broadcom Email Security

Broadcom Email Security is an enterprise-grade, high-accuracy email filtering solution designed to detect and block spam, phishing, and malware.

Documentation

BROADCOM

Source

Broadcom ESX

The primary function of ESX is to partition a physical server into multiple virtual machines (VMs), allowing different operating systems to run simultaneously and improving hardware utilization.

Documentation

BROADCOM

Source

Broadcom NSX

Provides network virtualization, micro-segmentation, and security for software-defined data centers.

Documentation

BROADCOM Source

Broadcom SSL Visibility Appliance

Broadcom's SSL Visibility (SSLV) Appliance is a dedicated network security device that decrypts, inspects, and re-encrypts SSL/TLS traffic to expose threats hidden within encrypted communications.

Documentation

BROADCOM Source

Broadcom vCenter

vCenter is a centralized management platform that provides a single console for managing an entire VMware vSphere virtual infrastructure, including multiple ESXi hosts and virtual machines.

Documentation

CHECK POINT Source

Check Point Software Anti-Bot

Detects and blocks botnet communications and command-and-control traffic to prevent malware infections.

Documentation

CHECK POINT Source

Check Point Software Anti-Malware

Protects endpoints from viruses, ransomware, and other malware using signature and behavior analysis.

Documentation

CHECK POINT Source

Check Point Software Anti-Phishing

Prevents phishing attacks by analyzing email content and links to block credential theft attempts.

Documentation

CHECK POINT Source

Check Point Software Anti-Spam and Email Security

Blocks spam and malicious email content using reputation checks and email filtering techniques.

Documentation

Quantum
Secure the Network

Source

Check Point Software Quantum

Unified threat prevention platform delivering firewall, VPN, and intrusion prevention capabilities.

Contact support

CHECK POINT

Source

Check Point Software CPMI Client

Legacy Check Point management client used to interface with security policies and logs.

Documentation

CHECK POINT

Source

Check Point Software cpmidu_update_tool

Utility used to update configuration and database files for Check Point Multi-Domain environments.

Documentation

CHECK POINT

Source

Check Point Software Database Tool

Command-line tool to extract, query, or update Check Point configuration and policy databases.

Documentation

CHECK POINT

Source

Check Point Software Endpoint Compliance

Checks endpoint status and posture before granting network access, enforcing security policies.

Documentation

CHECK POINT

Source

Check Point Software Endpoint Management

Centralized platform for managing endpoint protection, updates, and policy enforcement.

Documentation

CHECK POINT

Source

Check Point Software Next-Generation Firewall (NGFW)

Next-generation firewall providing intrusion prevention, application control, and threat protection.

Documentation

CHECK POINT

Source

Check Point Software Forensics

Analyzes security incidents on endpoints to uncover attack vectors and malicious activity.

Documentation

CHECK POINT

Source

Check Point Software Generic Checkpoint Product

Placeholder integration used for generic or unsupported Check Point log sources.

Documentation

CHECK POINT

Source

Check Point Software GO Password Reset

Facilitates secure password reset processes for users across integrated environments.

Documentation

CHECK POINT

Source

Check Point Software HTTPS Inspection

Decrypts and inspects HTTPS traffic to detect hidden threats within encrypted web sessions.

Documentation

CHECK POINT

Source

Check Point Software iOS Profiles

Provides configuration profiles for secure mobile access and web filtering on iOS devices.

Documentation

CHECK POINT

Source

Check Point Software IPS

Detects and blocks known and unknown exploits, malware, and vulnerabilities in network traffic.

Documentation

CHECK POINT

Source

Check Point Software MDS Query Tool

CLI tool for querying multi-domain configurations and policies in Check Point environments.

Documentation

CHECK POINT

Source

Check Point Software Media Encryption & Port Protection

Secures USB ports and encrypts removable media to protect sensitive data on endpoints.

Documentation

CHECK POINT

Source

Check Point Software Mobile Access

Enables secure remote access to corporate apps and data from mobile devices.

Documentation

CHECK POINT

Source

Check Point Software QoS

Implements bandwidth control and traffic prioritization policies for optimized network usage.

Documentation

CHECK POINT

Source

Check Point Software Query Database

Accesses and queries internal policy or object databases in Check Point systems.

Documentation

CHECK POINT

Source

Check Point Software SmartConsole

Graphical interface for managing Check Point security policies, logs, and monitoring.

Documentation

CHECK POINT

Source

Check Point Software SmartUpdate

Tool for updating and managing licenses, software, and hotfixes in Check Point environments.

Documentation

CHECK POINT

Source

Check Point Software Syslog Data

Enables export of Check Point logs via syslog to external monitoring and SIEM platforms.

Documentation

CHECK POINT

Source

Check Point Software Threat Emulation and Anti-Exploit

Emulates files in a virtual environment to detect and block advanced persistent threats and exploits.

Documentation

CHECK POINT

Source

Check Point Software URL Filtering

Controls and logs web access based on URL categories and custom site rules to enforce policy.

Documentation

CHECK POINT

Source

Check Point Software Web API

Provides programmatic access to Check Point security management through RESTful API endpoints.

Documentation

Source

Cisco Application Control Engine (ACE)

Provides application-aware load balancing, SSL offload, and traffic control for Cisco networks.

Documentation

Source

Cisco Access Control System (ACS)

Centralizes network access control with RADIUS and TACACS+ for authentication and authorization.

Documentation

Source

Cisco Adaptive Security Appliance (ASA)

Provides stateful firewall, VPN support, and advanced threat protection for secure network perimeters.

Documentation

Source

Cisco Catalyst

Delivers enterprise-grade Ethernet switching with high performance, security, and scalable management.

Contact support

Source

Cisco Integrated Management Controller (IMC)

Provides out-of-band server management for Cisco UCS, enabling hardware monitoring and configuration.

Documentation

Source

Cisco Digital Network Architecture (DNA)

Provides software-defined networking, policy automation, and analytics for enterprise infrastructure.

Documentation

Source

Cisco Duo

Delivers two-factor authentication and secure access controls to protect users and applications.

Contact support

Source

Cisco Email Security Appliance (ESA)

Protects email systems from spam, phishing, malware, and data loss with advanced threat filtering.

Documentation

Source

Cisco Firepower

Provides next-gen firewall features including intrusion prevention, app control, and malware protection.

Documentation

Source

Cisco Firepower Threat Defence (FTD)

Unifies firewall, VPN, and intrusion prevention into a single software for comprehensive threat defense.

Documentation

Source

Cisco Firewall Services Module (FWSM)

Delivers multi-context, high-performance firewall services integrated into Cisco Catalyst switches.

Documentation

Source

Cisco Internetwork Operating System

Network operating system for Cisco routers and switches, enabling routing, switching, and security.

Documentation

Source

Cisco Identity Services Engine (ISE)

Manages network access control and enforces policies with user and device authentication capabilities.

Documentation

Source

Cisco Meraki MX

Cloud-managed network appliance offering firewall, VPN, SD-WAN, and security in a single platform.

Documentation

Source

Cisco Meeting Management

Cisco Meeting Management is a management tool for Cisco Meeting Server, an on-premises video meeting platform.

Documentation

Source

Cisco Secure Firewall

Provides advanced threat protection and policy-based access control.

Contact support

Source

Cisco Private Internet eXchange (PIX)

Legacy firewall appliance delivering stateful inspection and secure network access control.

Documentation

Source

Cisco TelePresence Video Communication Server (VCS)

Enables video conferencing control and call routing for Cisco TelePresence systems and endpoints.

Documentation

Source

Cisco Unified Communications Manager (UCM)

Delivers unified voice, video, messaging, and mobility services in enterprise IP telephony systems.

Documentation

Source

Cisco HyperFlex (HX, UCSH)

Infrastructure solution combining compute, storage, and networking in a single system.

Documentation

Source

Cisco Unified Computing System Manager (UCSM)

Centralized management platform for Cisco Unified Computing System (UCS) servers and resources.

Documentation

Source

Cisco Viptela

Software-defined WAN solution providing secure connectivity, centralized control, and traffic optimization.

Documentation

Source

Cisco IOS XR

High-performance, modular network operating system for carrier-grade routing and scalability.

Documentation

Source

Citrix NetScaler

Offers application delivery, load balancing, and security features for optimized app performance.

Documentation

ClickHouse

Destination

ClickHouse

Columnar database optimized for real-time analytics on large datasets with high performance and scalability.

Documentation

coda

Source

Coda

All-in-one collaborative document platform combining text, spreadsheets, and integrations into a single workspace.

Contact support

COHESITY

Source

Cohesity DataProtect

Backup and recovery cluster that sends cluster audit, API access, data protection, and SMB file audit events.

Contact support

CONFLUENT

Destination

Confluent

Contact support

corelight

Source

Corelight NDR Platform

Provides network detection and response by analyzing traffic for advanced threats and anomalous behavior.

Documentation

zeek

Corelight Zeek

Contact support

CROWDSTRIKE

Destination

CrowdStrike Falcon LogScale

Cloud-native platform for endpoint detection, threat hunting, and security analytics at scale.

Contact support

CROWDSTRIKE

Destination

CrowdStrike Falcon Next-Gen SIEM

CrowdStrike Falcon Next-Gen SIEM unifies detection, investigation, and response on the Falcon platform. This destination sends logs to its HTTP Event Data Connector.

Documentation

CROWDSTRIKE

Source

CrowdStrike Falcon Platform

CrowdStrike Falcon is a cloud-native, AI-powered cybersecurity platform designed to stop breaches by protecting endpoints, cloud workloads, identity, and data.

Documentation

CYBERARK®

Source

CyberArk Privileged Threat Analytics (PTA)

Analyzes privileged account behavior to detect threats and suspicious activity in real time.

Documentation

CYBERARK®

Source

CyberArk Vault

Stores and manages privileged credentials, session recordings, and access control policies securely.

Documentation

databricks

Destination

Databricks

Unified analytics platform for data engineering, machine learning, and collaborative data science workflows.

Contact support

DATADOG

Destination

Datadog

Observability platform that provides monitoring for infrastructure, applications, logs, and security events.

Contact support

Source

DELL SonicWall

Delivers firewall, VPN, and deep packet inspection to protect networks from cyber threats and intrusions.

Documentation

Django

Contact support

Destination

Dynatrace

Unified Observability – Drive business transformation with contextual analytics, AI, and automation.

Documentation

Elasticsearch

Destination

Elasticsearch

Distributed search and analytics engine for real-time log analysis, observability, and data indexing.

Documentation

elastic

Source

Elasticsearch Bulk API

Bulk API endpoint that accepts events from Beats, Logstash, and other Elasticsearch clients.

Documentation

Everpure

Source

Everpure FlashArray

All-flash array whose Purity OS sends alerts, REST logins, CLI audit records, and capacity data over syslog.

Contact support

Source

F5 Networks BIG-IP

Provides load balancing, traffic management, and application security for optimized service delivery.

Documentation

Forcepoint Source

Forcepoint Email Security

Protects email systems from spam, phishing, malware, and data exfiltration using advanced threat defense.

Documentation

Forcepoint Source

Forcepoint NGFW

Next-gen firewall with deep packet inspection, policy enforcement, and integrated intrusion prevention.

Documentation

Forcepoint Source

Forcepoint Web Security

Provides web traffic filtering, malware protection, and data loss prevention for secure internet access.

Documentation

FORTINET Source

Fortinet FortiGate

Enterprise firewall platform offering threat protection, VPN, and traffic filtering for secure networking.

Documentation

FORTINET Source

Fortinet FortiMail

Secures inbound and outbound email with spam filtering, malware protection, and advanced threat detection.

Documentation

FORTINET

Fortinet FortiOS

Contact support

FORTINET

Source

Fortinet FortiProxy

Protects from internet threats like malware, phishing, and data loss by inspecting web traffic, filtering content, controlling applications, and enforcing security policies

Documentation

FORTINET

Source

Fortinet FortiWeb

Web application firewall protecting websites from attacks like XSS, SQL injection, and bot threats.

Documentation

FORTRA

Source

Fortra Powertech SIEM Agent for IBM i

Monitors IBM i system activity and forwards security events for centralized analysis in SIEM platforms.

Documentation

Source

Generic CEF Placeholder

A generic placeholder for fallback CEF classifications

Contact support

dev∅null

Destination

Generic /dev/null

Blackhole destination for testing or discarding events without storing them.

Documentation

Source

Generic HTTP

HTTP endpoint that accepts events from any client posting JSON or line-separated payloads.

Documentation

Source

Generic JSON Placeholder

A generic placeholder for fallback JSON classifications

Contact support

Source

Generic LEEF Placeholder

A generic placeholder for fallback LEEF classifications

Contact support

syslog

Source, Destination

Generic Syslog

Standard protocol for system logs. Our source automatically detects, parses, and classifies incoming syslog events without pre-configuration.

Documentation



Source

Generic Webhook

Receives data via HTTP POST requests for flexible event-driven integration.

Documentation

Google

Google Cloud

Contact support

Google

Source

Google Workspace Admin

Captures administrative events and configuration changes across the Google Workspace environment.

Contact support

Google

Source

Google Workspace Alert

Generates security and compliance alerts for suspicious user activity within Google Workspace services.

Contact support

Google

Destination

Google Pub/Sub

Asynchronous messaging service for ingesting and distributing event data between services and applications.

Documentation

Google

Destination

Google SecOps

A cloud service designed for enterprises to privately retain, analyze, and search the massive amounts of security and network telemetry they generate.

Documentation

Grafana loki

Destination

Grafana Loki

Scalable log aggregation system optimized for storing and querying logs by labels using Grafana.

Contact support

HAPROXY

Source

HAProxy

High-performance TCP/HTTP load balancer and reverse proxy for increasing reliability and scalability.

Contact support

Source

Hewlett Packard Enterprise Aruba Networks

Enterprise-grade networking gear for wireless, wired, and security management across distributed networks.

Documentation

Huawei

Contact support

Source

Idira

Identity security platform whose SIEM integration stream API delivers audit events for identity and access activity.

Documentation

Source

Imperva Incapsula

Cloud-based WAF, DDoS protection, and bot mitigation service for securing web applications and APIs.

Documentation

Source

Imperva SecureSphere

Provides on-prem web application, database, and file security with granular activity monitoring.

Documentation

Source

Infoblox NIOS

Delivers secure DNS, DHCP, and IPAM (DDI) services with centralized network control and automation.

Documentation

Source

Ivanti Connect Secure

Provides dynamic IP address assignment and network configuration for DHCP-enabled devices.

Documentation



Source

Juniper Networks Junos OS

Junos OS is the network operating system for Juniper physical and virtual networking and security products.

Documentation



JVM

Contact support



Destination

Kafka Apache Kafka

Distributed event streaming platform used for building real-time data pipelines and stream processing.

Contact support

kaspersky

Source

Kaspersky Endpoint Security

Protects endpoints from malware, ransomware, and intrusions with antivirus, firewall, and threat detection.

Documentation

NGINX
Part of F5

Source

Kubernetes Ingress NGINX Controller

Ingress NGINX Controller

Contact support

kubernetes

Source

Kubernetes Telemetry Controller

Telemetry Controller

Contact support



The Linux Foundation Linux

Contact support



Microsoft Azure

Contact support



Destination

Microsoft Azure Blob Storage

Cloud-based object storage optimized for unstructured data like backups, logs, and media files.

Contact support



Source

Microsoft Azure Event Hubs

Big data streaming platform to ingest and process events.

Documentation



Destination

Microsoft Azure Monitor

Collects and analyzes telemetry from applications and infrastructure to monitor performance and usage.

Documentation

Microsoft

Source

Microsoft Cloud App Security (MCAS)

Monitors cloud app usage, detects anomalies, and enforces security policies across SaaS services.

Documentation

GitHub

Source

Microsoft GitHub

Provides audit and event data for code repositories, development workflows, and security scans.

Contact support

Microsoft

Microsoft 365

Contact support

Microsoft 365

Source

Microsoft Office 365

Generates user activity and audit logs from Exchange, SharePoint, Teams, and other Office 365 services.

Contact support

Destination

Microsoft Sentinel

Cloud-native SIEM and SOAR platform for collecting, analyzing, and responding to security threats.

Documentation

Microsoft

Source

Microsoft Windows

Event logs from core services like security, system, DNS, and DHCP for operational and forensic analysis.

Documentation

Microsoft

Source

Microsoft Windows Event Log

Collects Windows Event Log data via OpenTelemetry Collector for centralized log analysis and monitoring.

Documentation

Microsoft

Source

Microsoft Windows Event Tracing

Collects Windows Event Tracing data via OpenTelemetry Collector for centralized log analysis and monitoring.

Documentation

Microsoft

Source

Microsoft Windows Events

Collects Windows Event Log data via WEC for centralized log analysis and monitoring.

Documentation

MIKROTIK

Source

MikroTik RouterOS

Router operating system providing firewall, bandwidth management, routing, and hotspot functionality.

Documentation

modsecurity
Open Source Web Application Firewall

ModSecurity

Contact support

Source

NetFlow Logic Optimizer

Aggregates and transforms flow data (NetFlow, IPFIX) into actionable security and performance insights.

Documentation

Source

Netgate pfSense

Open-source firewall and router platform with VPN, traffic shaping, and intrusion detection capabilities.

Documentation

Source

Netmotion

Provides secure, optimized remote access with performance monitoring for mobile and distributed workforces.

Documentation

NETSCOUT

Source

NETSCOUT Arbor Edge Defense (AED)

Edge-based DDoS protection and threat mitigation system to block attacks before they enter the network.

Documentation

NETSCOUT

Source

NETSCOUT Pravail (APS)

Monitors and mitigates advanced persistent threats and malware with inline packet inspection.

Documentation

Source

Generic UNIX/Linux host Generic Placeholder

A generic placeholder for program classifications

Contact support

Auth0 by Okta

Source

Okta Auth0

Identity-as-a-Service platform offering authentication and authorization for apps and APIs.

Contact support

Horizon® 8

Source

Omnissa Horizon

Horizon is a platform for managing and delivering virtual desktops and applications to end-users across a variety of devices and locations.

Documentation

onelogin

One Identity OneLogin

Contact support

OpenCanary

Contact support

OpenJS Foundation Node.js

Contact support

Destination

OpenObserve

High-performance log analytics platform supporting structured log ingestion, search, and visualization.

Documentation

Destination

OpenSearch

Open-source search and analytics suite for log ingestion, indexing, dashboards, and alerting.

Contact support

Source

OpenTelemetry File Collector

Collects observability data like traces, metrics, and logs from applications using the axoflow-otel-collector.

Documentation

Source

OpenTelemetry Journald Collector

Parses Journald events from systemd journal using the axoflow-otel-collector.

Documentation

Source

OpenTelemetry (OTLP)

Collects observability data like traces, metrics, and logs from applications using the OTLP protocol.

Documentation

Source

OpenText ArcSight

SIEM platform for collecting, correlating, and analyzing security event data across IT environments.

Documentation

Source

OpenText Self Service Password Reset (SSPR)

Allows users to securely reset their own passwords without IT assistance, reducing helpdesk load.

Documentation

Destination

Palo Alto Networks Cortex XSIAM

Extended Security Intelligence and Analytics Management (XSIAM) platform for advanced threat detection and response.

Documentation

Source

Palo Alto Networks Cortex XSOAR

Security orchestration, automation, and response platform for threat detection and incident management.

Documentation



Source

Palo Alto Networks Idira

Polls the Idira audit API for identity security platform events, authenticating with OAuth2 client credentials.

Documentation



Palo Alto Networks

Contact support



Source

Palo Alto Networks PAN-OS

Firewall operating system delivering network security features including traffic control and threat prevention.

Documentation



Source

Ping Identity PingAccess

A centralized access security solution which provides secure access to applications and APIs down to the URL level

Documentation

Source

Progress Flowmon ADS

Detects network anomalies and threats through flow-based behavior analysis and machine learning.

Documentation

Python Software Foundation Python

Contact support

Qualys

Contact support

riverbed

Source

Riverbed SteelConnect

Software-defined WAN solution for centralized network management and secure cloud connectivity.

Documentation

riverbed

Source

Riverbed SteelHead

WAN optimization appliance that accelerates application performance and reduces bandwidth usage.

Documentation

RPCFW

RPC Firewall

Contact support

RSA

Source

RSA Authentication Manager

Manages two-factor authentication using RSA SecurID tokens for secure access to enterprise resources.

Documentation

Ruby on Rails

Contact support

SECUREAUTH

Source

SecureAuth Identity Platform

Delivers identity and access management with adaptive authentication and single sign-on capabilities.

Documentation

servicenow

ServiceNow Computer

Computer configuration item discovered from ServiceNow CMDB.

Contact support

servicenow

ServiceNow Firewall Hardware

Hardware firewall configuration item discovered from ServiceNow CMDB.

Contact support

servicenow

ServiceNow IP Firewall

IP firewall configuration item discovered from ServiceNow CMDB.

Contact support

servicenow

ServiceNow IP Router

IP router configuration item discovered from ServiceNow CMDB.

Contact support

servicenow

ServiceNow IP Switch

IP switch configuration item discovered from ServiceNow CMDB.

Contact support

servicenow

ServiceNow Linux Server

Linux server configuration item discovered from ServiceNow CMDB.

Contact support

servicenow

ServiceNow Network Gear

Network gear configuration item discovered from ServiceNow CMDB.

Contact support

servicenow

ServiceNow Server

Server configuration item discovered from ServiceNow CMDB.

Contact support

servicenow

ServiceNow Windows Server

Windows server configuration item discovered from ServiceNow CMDB.

Contact support

servicenow

ServiceNow

IT service management platform used for host attribution.

Contact support

Skyhigh Security

Source

Skyhigh Security Secure Web Gateway

Inspects and filters web traffic to protect against malware, enforce policies, and prevent data loss.

Documentation

slack

Source

Slack

Collaboration platform that captures workspace messages, file uploads, and user activity for auditing.

Contact support

Destination

Snowflake

Cloud data platform for storing, querying, and analyzing large-scale structured and semi-structured data.

Contact support

Destination

Splunk

Ingests, indexes, and visualizes data for monitoring, analysis, and alerting.

Documentation

Source

Splunk HEC Server

HTTP Event Collector endpoint that accepts events from Splunk forwarders and HEC clients.

Documentation

Source

Splunk (as source)

Receive data from Splunk.

Documentation

Splunk Universal Forwarder

Agent that collects and forwards logs from remote systems to a central Splunk instance.

Contact support

Spring

Contact support

SQL

Contact support

Sumo Logic

Destination

Cloud-native platform for log management, metrics analysis, and security analytics.

Documentation

Source

Superna Eyeglass

Manages and automates data protection, DR, and reporting for PowerScale environments.

Documentation

Source

Tanium Platform

Endpoint management and security software that provides organizations with visibility and control across their IT environments.

Documentation

Source

Tenable Vulnerability Management

Cloud-based vulnerability management platform whose audit log API records user and configuration activity.

Documentation

THALES

Source

Thales Vormetric Data Security Platform

Provides data encryption, key management, and access controls across cloud and on-premise environments.

Documentation

Trellix

Source

Trellix Central Management System (CMS)

Centralized policy and configuration management platform for Trellix security products.

Documentation

Trellix

Source

Trellix ePolicy Orchestrator (EPO)

Manages endpoint security policies and threat responses across distributed environments.

Documentation

Trellix

Source

Trellix Email Threat Prevention (ETP)

Analyzes and filters email traffic to block phishing, malware, and targeted email-based threats.

Documentation

Trellix

Source

Trellix Generic

Contact support

Source

Trellix Endpoint Security (HX)

Detects and responds to advanced threats on endpoints using behavior-based analysis and threat intel.

Documentation

Source

Trellix Intrusion Prevention System

Trellix Intrusion Prevention System is an NDR-ready, next-generation IPS that detects and blocks sophisticated malware threats across the network.

Documentation

Source

Trellix MPS

Appliance for detecting and blocking advanced threats through inline malware inspection.

Documentation

Source

Trend Micro Deep Security Agent

Provides anti-malware, intrusion prevention, and log inspection for cloud and on-prem servers.

Documentation

Source

Ubiquiti Unifi

Manages network devices including routers, switches, and access points with centralized control.

Documentation

Source

Varonis DatAdvantage

Monitors data access and permissions to detect insider threats and automate compliance reporting.

Documentation

Source

Vectra AI AI Platform

Detects and investigates cyberattacks across cloud, data center, and enterprise networks using AI.

Documentation

Source

VMware ESXi

Bare-metal hypervisor that enables server virtualization and resource allocation for virtual machines.

Documentation

Source

VMware vCenter

Centralized management platform for VMware vSphere environments and virtual infrastructure control.

Documentation

Source

Wiz

Cloud security platform offering visibility into misconfigurations, vulnerabilities, and compliance risks.

Contact support

Source

Zscaler Internet Access

Cloud-based secure internet gateway that inspects traffic for threats and enforces policies.

Documentation

Source

Zscaler ZScaler Private Access

Provides secure remote access to internal apps without exposing them to the public internet.

Documentation

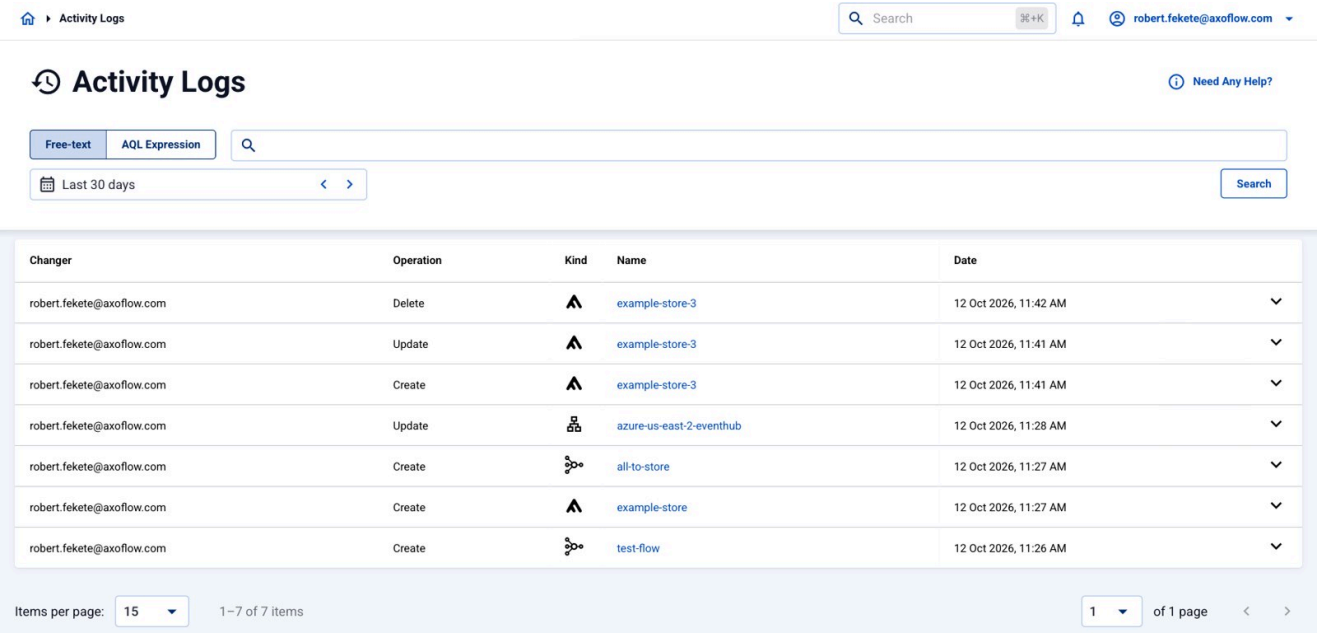
14 Activity log

Track and audit the configuration changes your team makes in AxoConsole, including who changed what and when, with a diff of each change.

Axoflow collects an activity log to help you track and audit the configuration changes and modifications done by your team.

> **Note**
>
> **Log tapping isn't recorded in the activity log by default. To include log tapping in the activity logs, contact our support team.**

To access the activity log, select **Activity Logs** in the main menu.

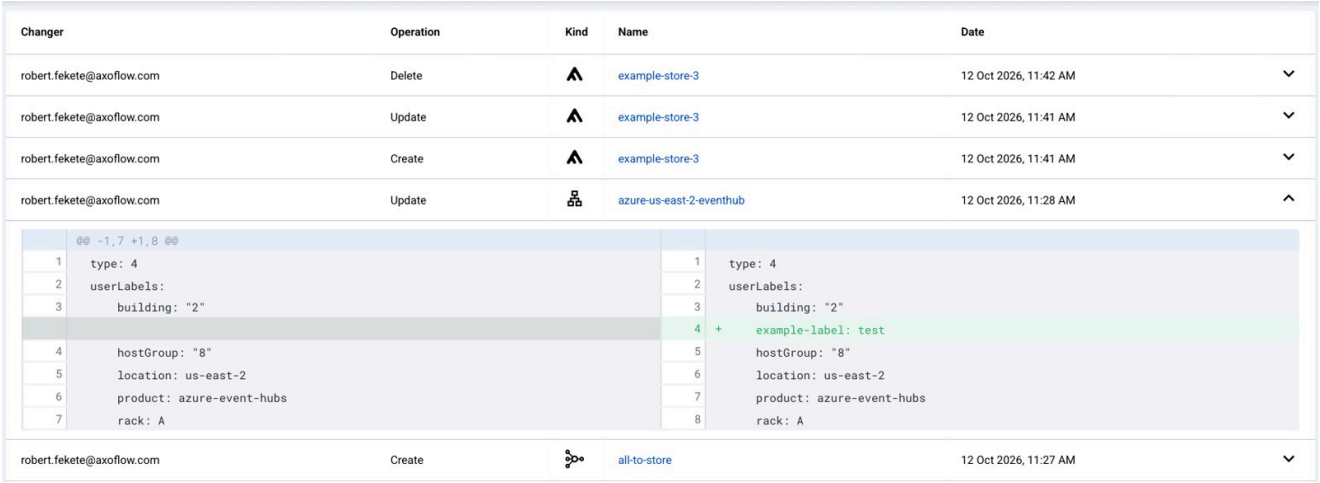


For each activity, the following information is shown:

- **Changer**: The user who performed the change.
- **Operation**: The type of the configuration change, one of: **Create**, **Update**, or **Delete**.
- **Kind**: The type of the configuration object that was modified: `Connector Rule`, `Destination`, `Flow`, `Host`, `Host Registration Request`, `Rehydration`, `Role`, or `Store`.
- **Name**: The name of the object that was modified.
- **Date**: The date when the change occurred.

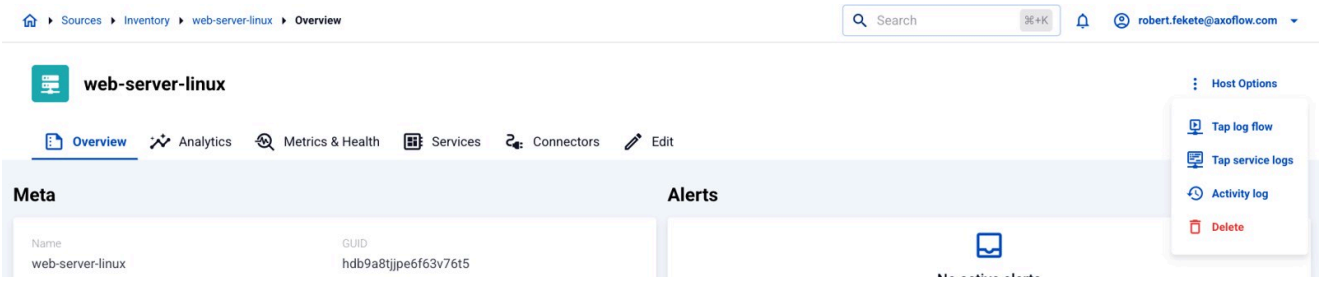
  Axoflow stores all dates in Coordinated Universal Time (UTC), and automatically converts it to the timezone of set in your browser/operating system.

Select the arrow at the end of a row to show a diff of what changed. The diff doesn't show sensitive data like tokens and passwords.



Filter activities

Use the filter bar to search for specific activities, and to set the time range of the activities. You can also open the activity log of a specific object: select **Activity log** in the menu of a host (**Host Options** on the **Sources** and **Routers** pages), a destination, a flow, or a store. In this case, the list shows only the activities of that object.



The settings of the filter bar change the URL parameters of the page, so you can bookmark it, or share a specific view by sharing the URL.

- **Free-text** mode searches in the following fields of the activity logs: **Changer**, **Operation**, **Kind**, **Name**.

  The search is case insensitive. Adding multiple keywords searches for matches in any of the previous fields. This is equivalent to the `@ANY =* keyword1 AND @ANY =* keyword2` AQL query.

- Use **AQL Expression** mode to search in specific fields of the activity logs.

  It also makes more complex filtering possible, using the Equals, Contains (partial match), and Match (regular expression match) operators. Note that:

  - To execute the search, click **Search**, or hit ESC then ENTER.

- AxoConsole autocompletes the built-in and custom labels and field names, as well as their most frequent values, but doesn't autocomplete labels and variables created by data parsing and processing steps.
- You can use the AND and OR operators to combine expressions, and also parenthesis if needed. For details on AQL, see AQL operator reference.
- The precedence of the operators is the following: parentheses, AND, OR, comparison operators.
- Use the usual keyboard shortcuts to undo (⌘/Ctrl + Z) or redo (⌘/Ctrl + Shift + Z) your edits.

15 Troubleshooting Axoflow

Find the cause of a problem from what you see, like a source that doesn't show up or data that doesn't reach the destination.

Find the symptom that matches your problem, then complete the checks in order. The checks use the following tools of AxoConsole:

- Log tapping shows samples of the data that a host receives and sends. To see the internal logs of the services on a host, tap the service logs.
- Flow tapping shows how the processing steps of a flow change the data.
- Alerts show the problems that Axoflow detected in your pipeline.
- Host metrics show the history of the connections, the throughput, the queues, and the dropped messages of a host.
- On the AxoRouter host, the `axorouter-ctl` command shows the logs and the statistics of AxoRouter.

A source doesn't appear on the Topology page

1. Check that the source sends its data to an AxoRouter instance that's registered in AxoConsole. A source appears in the **Detected** list only after AxoRouter receives data from it. To add the source, see onboard a vendor source.
2. Check that the source sends its data to a port and protocol that a connector of the AxoRouter accepts. For the default ports, see the open ports. To receive data on other ports, see AxoRouter connector rules.
3. Check that the firewall of the AxoRouter host allows traffic on the port of the connector.
4. Check that you configured the source as its page in the Vendors section describes. Other message formats or ports can cause data loss or incorrect parsing.
5. Tap the service logs of the AxoRouter, and look for errors about the connector, for example, TLS handshake errors.

AxoRouter receives the data, but it doesn't reach the destination

1. Check that a flow sends the data to the destination:

- The **Router Selector** of the flow matches the AxoRouter instance that receives the data.
- The flow is enabled.

For details, see Flow overview.

2. Check that the processing steps of the flow don't filter out the data, for example, a **Select Messages** step. To find the step that drops the data, add **Probe** steps to the flow, and check the flow metrics, or use Flow tapping.
3. If no other flow processes the data, check whether a fallback flow sends it to a different destination.
4. Check that the destination accepts the data. See The destination rejects the data.

The destination rejects the data

Check the logs of the AxoRouter that sends data to the destination:

1. Select **Topology**, then select your AxoRouter instance. Alternatively, select `⌘/Ctrl + K` and enter the name of your AxoRouter.
2. Select ⋮ **Host Options > Tap service logs**.
3. (Optional) To tap only the logs of a specific service, set the **Service name** field.

   Depending on the configuration of the host, the following services can be available:

   - **axoflow-cloudconnector**: The service that collects logs from cloud services and APIs.
   - **axoflow-otel-collector**: The collector agent on edge hosts.
   - **axorouter-logstore**: The service that handles Logstore type stores.
   - **axolet**: The monitoring and management agent for Axoflow pipeline elements.
   - **axorouter-resolver**: A service for retrieving host data from external CMDB systems.
   - **axorouter-syslog**: The main processing element of AxoRouter deployments.
   - **axorouter-wec**: The service that handles the Windows Events connector (WEC).
   - **axostore**: The service that handles AxoStore on the host. Only available if there are stores configured on the host.
4. If systemd is available on the host, select **Systemd service output** to show the logs of the service from the systemd journal. Otherwise, the **Internal logs** of the service are available.
5. Select **Start**. Axoflow displays the log messages of AxoRouter. Check the logs for error messages.

The following sections list common errors of specific destinations.

Splunk

Some common errors include:

- `Redirected event for unconfigured/disabled/deleted index=netops with source="source::axo" host="host::axosyslog-almalinux" sourcetype="sourcetype::fortigate_event" into the LastChanceIndex. So far received events from 1 missing index(es).` : The Splunk index where AxoRouter is trying to send data doesn't exist. Check which index is missing in the error message and create it in Splunk. (For a list of recommended indices, see the Splunk destination prerequisites.)
- `http: error sending HTTP request; url='https://prd-p-sp2id.splunkcloud.com:8088/services/collector/event/1.0?index=&source=&sourcetype=', error='SSL peer certificate or SSH remote key was not OK', worker_index='0', driver='splunk--flow-axorouter4-almalinux#0', location='/usr/share/syslog-ng/include/scl/splunk/splunk.conf:104:3'` : Your Splunk deployment uses an invalid or self-signed certificate, and the **Verify server certificate** option is enabled in the Splunk destination of Axoflow. Either fix the certificate in Splunk, or: select **Topology > <your-splunk-destination>**, disable **Verify server certificate**, then select **Update**.

Google Security Operations (SecOps)

If your logs aren't showing up in Google SecOps, tap the service logs of the axorouter-syslog service on an AxoRouter that's sending data to Google SecOps.

- If you see errors like `http: Server returned with a 4XX (client errors) status code, which means we are not authorized or the URL is not found;` , double-check the **Service endpoint** (make sure that it points to the right regional service endpoint) and **Customer ID** fields.

AxoRouter drops messages

1. Check the alerts of the AxoRouter host. Axoflow raises alerts when a host drops packets or messages, and when its disk queues fill up.
2. Check the **Dropped packets (total)** chart and the queue charts in the metrics of the host:
    - **Dropped UDP packets** means that the operating system dropped UDP packets before AxoRouter could process them.

- **Dropped log events** means that the event queues of AxoRouter dropped messages.
- Growing memory and disk queues mean that a destination doesn't accept the data as fast as AxoRouter receives it. Check the errors of the destination. See The destination rejects the data.

3. If the host drops UDP packets, increase the throughput of the connector that receives them. UDP has no flow control: when the packets of a connector arrive faster than AxoRouter can process them, for example, during a burst of traffic from a firewall, the operating system drops the packets that don't fit into the receive buffer. This can happen even if the host has free CPU capacity, because by default a single thread receives and processes the messages of a connector. A larger receive buffer absorbs short bursts, but doesn't help when the sustained rate is higher than what a single thread can process.

   Edit the connector:

   - Check the **Socket buffer size** of the connector and the `net.core.rmem_max` sysctl on the host. The recommended value is 32 MiB.
   - Set the number of **Processing workers** to process the messages of the connector on multiple threads.
   - If **UDP socket count** is higher than 1, enable **EBPF load balancing**. Otherwise, every message from a sender goes to the same socket, so a single high-volume sender doesn't benefit from the additional sockets. For details, see UDP load balancing.
   - If the source can send its data over TCP, use a TCP connector instead. Over TCP, AxoRouter slows the sender down instead of dropping data.

Disk is full on AxoRouter, or a service fills the disk with logs

1. Check what uses the space. Run `df -h /var /run`, `journalctl --disk-usage`, and `sudo du -xsh /var/log/* | sort -h` on the host.
2. If the logs use the space, find the service that writes them, free disk space, and limit the logs of the service. For details, see A service fills the disk with logs.
3. If the logs don't use the space, check the disk queue charts in the metrics of the host. Growing disk queues mean that a destination doesn't accept the data as fast as AxoRouter receives it. See The destination rejects the data.

An edge host doesn't send data

1. Check that a collection rule applies to the host: the **Edge selector** of the rule must match the labels of the host. For details, see add an edge host to a collection rule.
2. Check that a data forwarding rule sends the data of the host to an AxoRouter instance.
3. Check that the `axolet` and `axoflow-otel-collector` services are running on the host. For details, see Linux or Windows.

A host is unavailable, or its configuration doesn't update

1. Check the alerts of the host. Axoflow raises alerts when a host becomes unavailable, and when the configuration of a managed host fails to synchronize or reload.
2. Check that the `axolet` service is running on the host. For details, see Linux or Windows.
3. If you cloned the host from another host that runs Axolet, or the root CA certificate of AxoConsole changed, reinstall Axolet. For details, see Reinstall Axolet.

Installation problems

AxoConsole on premises

- If you get the `[ERROR] Timed out waiting for k3s node to be Ready` error message, increase the timeout limit by running `K3S_TIMEOUT=180`, then rerun the installation script.

In case of other errors, run the following command to create a debug package from the cluster, and contact our support team.

```
sudo k3s kubectl cluster-info dump --all-namespaces  --output-directory=./cluster-dump && tar -czvf cluster-dump.tar.gz cluster-dump
```

Axoflow agent for Windows

If you run into problems when installing Axoflow agent for Windows, complete the following steps.

1. Double-check that you're running the installation command:
   - as administrator,
   - in PowerShell, not in Cmd.
2. Set the following environment variable to store the installation logs. Adjust the directory as needed.

```
$env:TEMP_DIR = "C:\Temp\axoflow-agent-install"
```

3. Re-run the installation command.
4. Open the Windows Event Viewer and check the **Windows Logs > System** events at the time of installation for errors or issues logged by the Service Control Manager.
5. Check that the installation logs of `axolet` and `axoflow-otel-collector` are available, for example, the `C:\Temp\axoflow-agent-install\axolet-windows-amd64.msi-install.log` and `C:\Temp\axoflow-agent-install\axoflow-otel-collector_<version-number>-axoflow.1_windows_x64.msi-install.log` files exist.
6. If you suspect that the installer is incorrectly reporting connectivity issues, disable the preflight check and re-run the installation command.

```
$env:SKIP_PREFLIGHT_CHECK = 'true'
```

7. If needed, contact our support team.

Contact support

If you can't solve the problem, contact the Axoflow support team.

15.1 A service fills the disk with logs

Find the service that floods the logs, free disk space, rate-limit the service in the systemd journal, and cap the journal and syslog file size.

If a service writes log messages faster than usual, the systemd journal or the plain syslog file can grow until the disk fills up. This page shows how to:

- find the service that writes the logs,
- free disk space,
- limit how many log messages the service can write to the journal,
- keep those messages out of the plain syslog file, and
- cap the total size of the journal so one service can't fill the disk.

The examples use the Cloud connector on an AxoRouter host. It runs as the `axoflow-cloudconnector.service` systemd unit. Other AxoRouter services work the same way, for example, `axorouter.service`, `axostore.service`, or `axorouter-wec.service`. Set the `UNIT` and `PROGRAM` variables in the commands to the values of the service.

All commands need root privileges.

> **CAUTION:**
>
> Don't edit the unit file of a service that an Axoflow package installs. For example, systemd generates `axoflow-cloudconnector.service` from the Podman Quadlet file `/etc/containers/systemd/axoflow-cloudconnector.container`, which the AxoRouter package installs. The next AxoRouter upgrade replaces this file and your change is lost. Use a systemd drop-in file instead: upgrades don't touch drop-ins.

Find the full filesystem and the noisy service

1. Check which filesystem is full, and how much space the logs use:

   ```
   df -h /var /run                      # which filesystem is full
   journalctl --disk-usage              # space used by the journal
   sudo du -xsh /var/log/* | sort -h    # space used by plain log files
   ```

   Check `/var/log/syslog` (Debian, Ubuntu) or `/var/log/messages` (RHEL, Rocky Linux) in particular.

2. Find the services that logged the most messages in the last hour:

```
sudo journalctl --since -1h -o json --output-fields=_SYSTEMD_UNIT \
  | grep -o '"_SYSTEMD_UNIT":"[^"]*"' | sort | uniq -c | sort -rn | head
```

3. Check the recent messages of the service at the top of the list:

```
UNIT=axoflow-cloudconnector.service
journalctl -u "$UNIT" --since -1h | wc -l   # messages in the last hour
journalctl -u "$UNIT" -n 50                 # recent messages
```

   The messages often show the cause of the log volume, for example, a connection error that repeats.

Why the disk fills up

With its default settings, the journal limits its own size. It uses at most 10% of the filesystem (never more than 4 GiB), and leaves at least 15% of the filesystem free (never more than 4 GiB). If the disk is full anyway, the cause is usually one of these:

- **A plain-file copy of the logs.** rsyslog, syslog-ng, or AxoSyslog also writes the same messages to `/var/log/messages` or `/var/log/syslog`. This is the most common cause. The journal settings don't limit the size of this copy. To fix it, keep the service's messages out of the syslog file.
- **Increased journal size limits.** Someone raised `SystemMaxUse=` in `journald.conf`. To fix it, cap the size of the journal.
- **The service writes its own log file** instead of logging to stdout or stderr. The journal settings and the rate limit don't apply to this file.
- `Storage=volatile` **is set in** `journald.conf`. journald keeps the journal in `/run` (RAM), so it fills up memory instead of the disk.

Free disk space

If the disk is already full, run these commands to rotate the journal and shrink it to 500 MB.

CAUTION:

The `truncate` command empties the whole syslog file. This also deletes the logs of all other services on the host.

```
sudo journalctl --rotate                # close the active journal files so that
vacuum can remove them
sudo journalctl --vacuum-size=500M      # or: --vacuum-time=2d
# If the syslog file uses the space, empty it in place.
# Don't delete it with rm: the daemon keeps the deleted file open, and the space
isn't freed.
sudo truncate -s 0 /var/log/syslog      # or /var/log/messages
```

Rate-limit the logs of a service in the journal

The `LogRateLimitIntervalSec` and `LogRateLimitBurst` options of a systemd unit replace the global journald rate limit for that unit. The unit can write at most `LogRateLimitBurst` messages in each `LogRateLimitIntervalSec` interval. journald drops the excess messages.

1. Create a drop-in file that lets the unit write about 1000 messages every 30 seconds to the journal.

   CAUTION:

   The new limit only takes effect after a restart, which briefly interrupts the service. For the Axoflow Cloud Connector, collection from your cloud sources stops for a few seconds.

   ```
   UNIT=axoflow-cloudconnector.service
   sudo mkdir -p "/etc/systemd/system/$UNIT.d"
   sudo tee "/etc/systemd/system/$UNIT.d/50-log-ratelimit.conf" <<'EOF'
   [Service]
   # Keep at most 1000 messages per 30s from this unit; journald drops the rest
   LogRateLimitIntervalSec=30s
   LogRateLimitBurst=1000
   EOF
   sudo systemctl daemon-reload
   sudo systemctl restart "$UNIT"
   ```

   You can also create the drop-in file interactively with `sudo systemctl edit <service-name>`.

2. Check that the limit is active.

```
UNIT=axoflow-cloudconnector.service
systemctl status "$UNIT" | grep -A1 Drop-In   # lists 50-log-ratelimit.conf
systemctl show "$UNIT" -p LogRateLimitIntervalUSec -p LogRateLimitBurst
journalctl -u systemd-journald --since -10min | grep -i "suppressed.*$UNIT"
```

   The `systemctl show` command prints the values you set.

   - **If the values are empty,** the host runs systemd older than version 240, which doesn't support these options. Check the version with `systemctl --version`. In that case, set the global `RateLimitIntervalSec=` and `RateLimitBurst=` options in `/etc/systemd/journald.conf.d/` instead (defaults: `30s` and `10000`). The global options apply to every service on the host.
   - **If the `grep` prints `Suppressed N messages` lines,** the limit is working and journald is dropping the excess messages. journald prints this line when a new 30-second window starts and the service is still logging, so wait at least a minute before checking.

   More messages can get through than the configured burst. journald raises the burst when the journal has a lot of space available. For example, with 4 GiB available, about 4000 messages get through in each 30-second window.

If you want to remove the rate limit, see Remove the rate limit.

Keep the service's messages out of the syslog file

On Debian and Ubuntu, rsyslog receives messages from journald through the syslog socket. journald forwards these messages *before* it applies the rate limit. The journald rate limit therefore doesn't reduce what rsyslog writes to `/var/log/syslog`. On RHEL and Rocky Linux, rsyslog reads the journal (`imjournal`), so it gets only the messages that pass the rate limit.

If rsyslog runs on the host (`systemctl is-active rsyslog`), add a rule that stops it from writing the messages of the service. The messages stay available in the journal, with the rate limit applied.

1. Find the program name of the service. rsyslog uses the `SYSLOG_IDENTIFIER` field of the journal as the program name. For the Axoflow Cloud Connector, it is `axoflow-cloudconnector`.

```
UNIT=axoflow-cloudconnector.service
journalctl -u "$UNIT" -n 1 -o verbose | grep SYSLOG_IDENTIFIER
```

2. Add the rule, and restart rsyslog.

```
PROGRAM=axoflow-cloudconnector
sudo tee "/etc/rsyslog.d/10-drop-$PROGRAM.conf" <<EOF
# $PROGRAM logs to the systemd journal; keep it out of the syslog file
:programname, isequal, "$PROGRAM" stop
EOF
sudo rsyslogd -N1                # check the configuration
sudo systemctl restart rsyslog
```

Cap the size of the journal and the syslog file

Limiting the total size of the journal protects the disk, no matter which service produces the logs. Do this even if you already rate-limited the service.

```
sudo mkdir -p /etc/systemd/journald.conf.d
sudo tee /etc/systemd/journald.conf.d/50-size.conf <<'EOF'
[Journal]
SystemMaxUse=2G
SystemKeepFree=4G
EOF
sudo systemctl restart systemd-journald
```

Also make sure that `logrotate` rotates the syslog file by size. Edit the existing entry for the syslog file in `/etc/logrotate.d/rsyslog` (Debian, Ubuntu) or `/etc/logrotate.d/syslog` (RHEL, Rocky Linux), and add the `maxsize` and `rotate` options to it. Don't create a new file for the same log: logrotate reports an error for duplicate log entries. For example:

```
/var/log/syslog
{
    maxsize 1G
    rotate 4
    # keep the other options of the existing entry
}
```

Check that the disk usage stopped growing

Run these commands again an hour later, and compare the results:

```
journalctl --disk-usage
df -h /var
sudo du -xsh /var/log/* | sort -h
```

Remove the rate limit

To remove the rate limit of the service and the rsyslog rule, run the following commands.

```
UNIT=axoflow-cloudconnector.service
PROGRAM=axoflow-cloudconnector
sudo rm "/etc/systemd/system/$UNIT.d/50-log-ratelimit.conf"
sudo systemctl daemon-reload
sudo systemctl restart "$UNIT"
# If you added the rsyslog rule:
sudo rm "/etc/rsyslog.d/10-drop-$PROGRAM.conf"
sudo systemctl restart rsyslog
```

Limitations of the rate limit

- **Dropped messages are lost permanently.** They can include the error that explains the problem. Use the rate limit as a safety valve, not as a fix. The real fix is to remove the cause of the log volume, for example, to lower the log level of the service, or to upgrade AxoRouter. After that, you can raise or remove the limit.

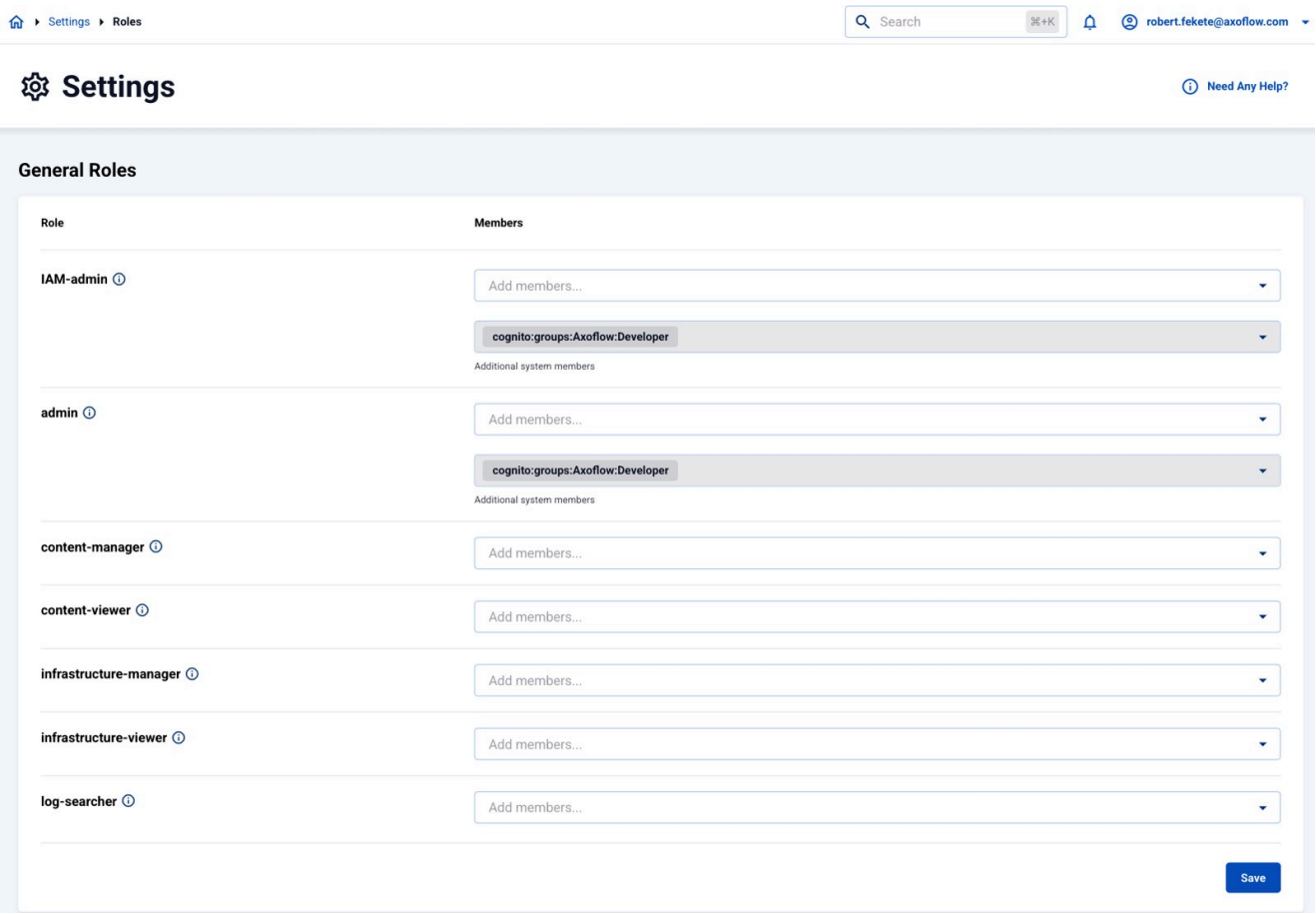
- **The limit only applies to messages that go through journald.** This includes stdout and stderr, `syslog()` calls, and containers that use the `journald` log driver. It doesn't apply to a log file that the service writes itself.

16 Access management

Control who can access AxoConsole by assigning roles to users based on group memberships, email domains, and individual email addresses.

Axoflow users can have different levels of access to AxoConsole based on their roles.

Authorization can be based on group memberships, email domains, and individual email addresses.



Roles

The following roles are available in AxoConsole by default:

- **IAM-admin**: Can manage the roles and permissions to access AxoConsole. Only this role can make changes on the **Settings > Roles** page. If your identity provider has an administration page, this role can also see the **Identity Provider Admin Panel** link to it on the **Settings > Roles** page. (If your identity provider has a profile page, every user can open their own profile by selecting **<their username> > Profile** in the top-right corner.)

- **admin**: Has full access to AxoConsole, except:
  - has only read access to the **Settings > Roles** page,
  - can't search logs in stores (that requires the **log-searcher** role, or access role to a specific store).
- **infrastructure-manager**: Can manage the infrastructure (without the permissions to view or tap log contents, or to rehydrate data). Has full access to the following pages: **Activity Logs**, **Alerting**, **Routers**, **Sources**, **Flows**, **Provisioning**, **Search Logs**. Can tap into service logs which may contain event payload.
- **infrastructure-viewer**: Similar to **infrastructure-manager**, but can only view the pages.
- **content-manager**: Can view log content and manage content related details like flows (without the permissions to manage infrastructure, but including access to view infrastructure details).
  - Has full access to the **Rehydration** page.
  - Can view and modify **Flows**, but can't create or delete them.
  - Can view **Routers**, **Sources**, **Search Logs**, **Analytics**.
  - Can tap into logs.
- **content-viewer**: Can view content like analytics, log search, log tapping (without the permissions to manage or view infrastructure details).
  - Can view **Search Logs**, **Analytics**.
  - Can tap into logs.
- **log-searcher**: Can search in every store. You can also grant access to specific stores.

If you need other roles, contact the Axoflow support team. Composing other roles is possible as part of a custom integration.

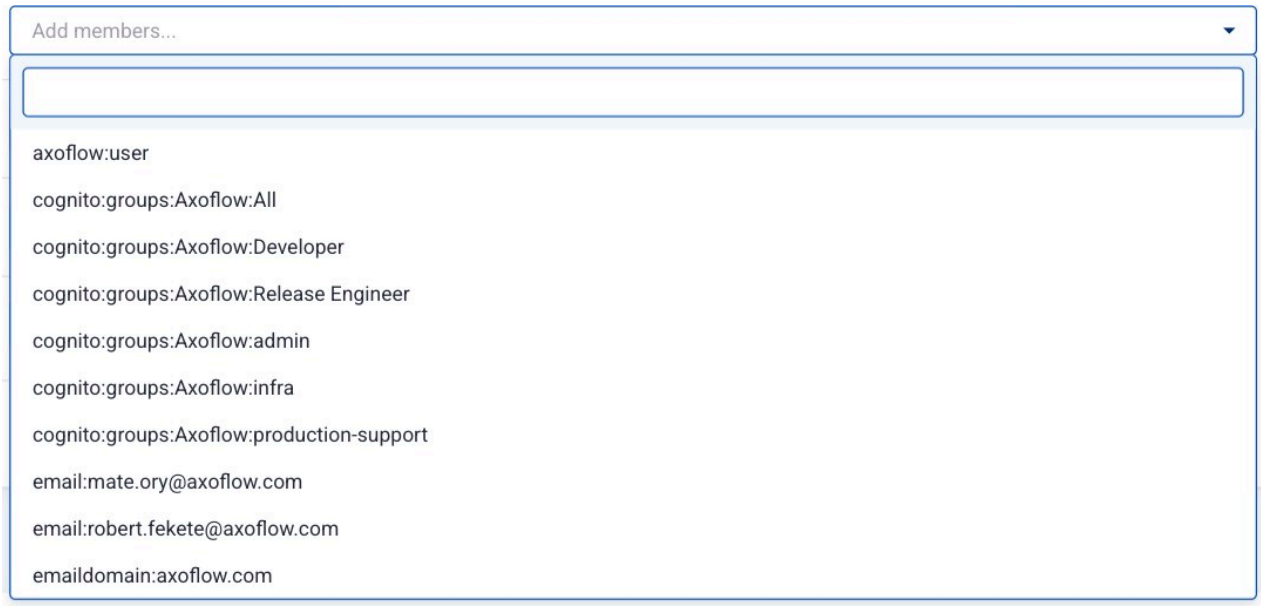
Manage access

To add new members to a role, complete the following steps. You need the **IAM-admin** role to manage the roles.

1. Navigate to **Settings > Roles**.
2. In the **General Roles** list, select the **Add members...** field of the role, and select or type the member. You can add:
   - A specific email address, for example, `email:username@example.com`

> **Note**
>
> - The user must be able to authenticate to AxoConsole using this email address, so authentication must be properly set up.
> - Multiple identity providers may allow authentication with the same email address, possibly with different level of ownership validation.

- An entire email domain, for example, `emaildomain:example.com`. Any user who authenticates with an email address belonging to this email domain will have access to the role.
- A user group. The format of the group depends on the OID provider.
  - If you're using AxoConsole as a SaaS, specify the group in the following format: `cognito:groups:<groupname-retrieved-from-cognito>`, for example, `cognito:groups:ExampleGoogleSaml:Operator`.
  - If you're using an on-prem AxoConsole deployment, specify the group in the following format: `groups:<groupname-retrieved-from-dex>` for example, `groups:operator`.
- A specific session of an authenticated user, in the following format: `user:<oidc-response-subject>`, for example: `user:42c4e962-f077-705f-138f-f01ba1220c44`.
- Every authenticated user: `axoflow:user`



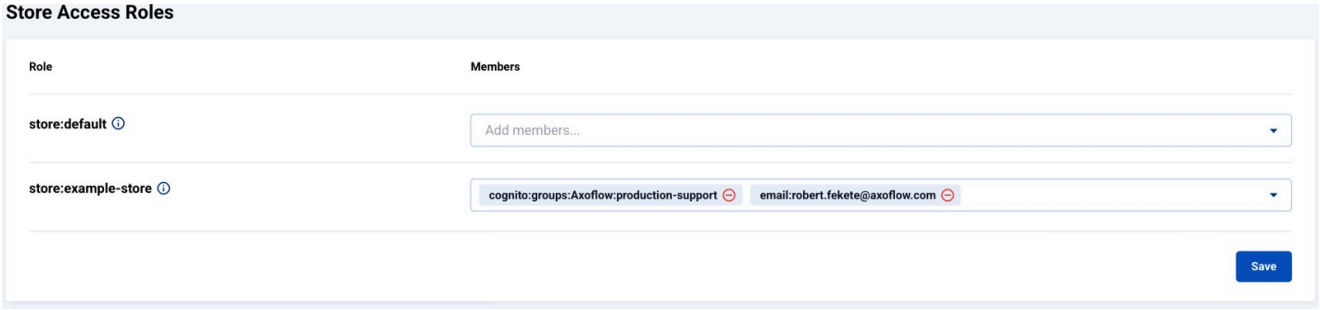
3. Select **Save**.

Some members are fixed: the **Additional system members** field of the role lists them, and you can't remove them. They were added during the integration at the request of your organization, or during the initial configuration of on-premises deployments.

Store access roles

The **Store Access Roles** grant search access to a specific store. The **Store Access Roles** section of the **Settings > Roles** page lists every configured store. To grant search access to a specific store, add the group or user to the **Members** of the store, then select **Save** in the **Store Access Roles** section. To grant access to every store, use the **log-searcher** role instead.

> **CAUTION:**
>
> These access roles have effect only when the user searches the store using AxoConsole. It doesn't have any effect if someone gains direct access to the store files (for example, when accessing the stores on an AxoRouter from a terminal).



17 Deploy AxoConsole

Deploy AxoConsole as a SaaS service, on premises, or on Kubernetes.

This chapter describes the installation and configuration of AxoConsole in different scenarios.

AxoConsole is available as:

- an SaaS service,
- on-premises deployment,
- Kubernetes deployment.

17.1 On-premise

Install, configure, authenticate, and upgrade an on-premises AxoConsole deployment, and prepare AxoRouter and syslog-ng hosts.

17.1.1 Deployment guide

Install AxoConsole on an on-premises virtual machine, prepare the AxoRouter hosts, and troubleshoot the installation.

This guide shows you how to install AxoConsole on a local, on-premises virtual machine using k3s and Helm. Installation of other components like the ingress controller and the cert-manager is also included. Since this is a single instance deployment, we don't recommend using it in production environments. For additional steps and configurations needed in production environments, contact our support team.

To deploy AxoConsole in an air-gapped environment, see Air-gapped.

> **Note**
>
> **AxoConsole is also available without installation as a SaaS service. You only have to install AxoConsole on premises if you want to run and manage it locally.**

At a high level, the deployment consists of the following steps:

1. Preparing a virtual machine.
2. Running the installation script on the virtual machine that deploys:
   - Kubernetes
   - cert-manager and Traefik
   - AxoConsole

   Any of these components can be skipped if already installed.
3. Basic authentication with admin user and password is configured by default. You can also configure other common authentication methods like LDAP, Github and Google.
4. Before deploying AxoRouter describes the steps you have to complete on a host before deploying AxoRouter on it. These steps are specific to on-premises AxoConsole deployments, and aren't needed when using the SaaS AxoConsole.

Prerequisites

To install AxoConsole, you'll need the following:

- The URL for the AxoConsole installation script. You'll receive this URL from our team. You can request it using the contact form.

  > **CAUTION:**
  >
  > Don't start the Install AxoConsole process until you've received the URL.

- A license key for AxoConsole. You'll receive this from our team. You can request it using the contact form.
- A host that meets the system requirements.
- Network access set up for the host.

System requirements

Supported platforms:

- Ubuntu 24.04 and 26.04,
- Red Hat Enterprise Linux 9.4 or newer and 10.

Other compatible and derivative distributions (for example, AlmaLinux or Rocky Linux) should work as well, but we don't test them explicitly.

The virtual machine (VM) must have at least:

Resource	Minimum	Production
CPU	4 vCPU	16 vCPU
RAM	8 GB	16 GB
Disk (`/`)	100 GB	250 GB

- A Minimum setup with 4 vCPU (x86_64-based), 8 GB RAM, and 100 GB disk space can handle about 100 AxoRouter instances and 1000 data source hosts.
- A real-life production scenario that handles 100 AxoRouter and 3000 data source hosts with 30-day metric retention would need:
  - 16 vCPU (x86_64-based)
  - 16 GB RAM
  - 250 GB disk space

> **Note**
>
> **Note that AxoRouter and Axoflow agent collects detailed, real-time metrics about the data-flows – giving you observability over the health of the security data pipeline and its components. Your security data remains in your self-managed cloud or in your on-prem instance where your sources, destinations, Axoflow agents, and AxoRouters are running, only metrics are forwarded to AxoConsole.**

For details on sizing, contact our support team.

You'll need to have access to a user with `sudo` privileges.

Network access

The host must be able to access the following domains related to the AxoConsole:

- When using AxoConsole SaaS:
  - `<your-tenant-id>.cloud.axoflow.io` : HTTPS traffic on TCP port 443, needed to download the binaries for Axoflow software (like Axolet and AxoRouter).
  - `kcp.<your-tenant-id>.cloud.axoflow.io` : HTTPS (mutual TLS) traffic on TCP port 443 for management traffic.
  - `telemetry.<your-tenant-id>.cloud.axoflow.io` : HTTPS (mutual TLS) traffic on TCP port 443, where Axolet sends the metrics of the host.
  - `us-docker.pkg.dev` : HTTPS traffic on TCP port 443, but only if you're forcing the installation script to pull the container images from the public repository. The Axolet and AxoRouter for Linux installation scripts download the images directly from AxoConsole.
- When using an on-premise AxoConsole:
  - The following domains should point to AxoConsole IP address to access Axoflow from your desktop and AxoRouter hosts:
    - `your-host.your-domain` : The main domain of your AxoConsole deployment.
    - `idp.your-host.your-domain` : A subdomain for the identity provider.
    - `kcp.your-host.your-domain` : A subdomain for management traffic.
    - `telemetry.your-host.your-domain` : A subdomain for agent telemetry.
  - The AxoConsole host must have the following **Open Ports**:
    - 80 TCP: Traefik ingress (HTTP, redirected to HTTPS).
    - 443 TCP: HTTPS for the AxoConsole UI, API, and all web traffic.
- When installing Axoflow agent for Windows or Linux:
  - `github.com` : HTTPS traffic on TCP port 443, for downloading installer packages.

Install AxoConsole

CAUTION:

This procedure covers how to deploy a new AxoConsole instance. The installation script can't upgrade existing deployments, contact the Axoflow support team for detailed upgrade instructions.

1. Download the installation script from the URL you've received from the Axoflow Support Team.
2. Transfer the file to the host where you want to install AxoConsole.
3. Run the installation script with your license key and the domain name for your AxoConsole deployment.

> **Note**
>
> **Re-running the installer will reuse any existing or modified manifests or user settings (for example, authentication settings added to `/var/lib/rancher/k3s/server/manifests/axoflow-config.yaml`) available on the host.**

```
sudo LICENSE_KEY=<your-license-key> BASE_HOSTNAME=<axoconsole-domain-name>
AXOFLOW_VERSION=0.90.0 bash ./axoflow-install.sh --no-airgap
```

> **Note**
>
> **If you omit `BASE_HOSTNAME`, the installer probes whether `<VM_IP>.nip.io` resolves and offers it as the default. nip.io is a free wildcard DNS service — handy for testing and demos because the console and all its subdomains resolve without any DNS or `/etc/hosts` configuration. For production deployments, use a proper domain name.**

You can use the following flags as needed for your environment.

- `--step`: Prompt before each step
- `--dry-run`: Show what would be done without executing anything
- `--skip-k3s`: Skip k3s installation (already installed)
- `--skip-cert-manager`: Skip cert-manager installation
- `--disable-traefik`: Disable the built-in Traefik ingress controller of k3s. Use this option when using a custom ingress controller.

- `--skip-nip-io` : Skip the automatic `<VM_IP>.nip.io` DNS probe and use `axoflow.local` as the default base hostname.
- `--diagnose [namespace]` : Inspect pods that are not ready and explain why (image-pull errors, configuration issues, crashes). Runs automatically on a timeout; use on demand after a failed install.
- `--uninstall` : Uninstall AxoConsole. Prompts before each step – removing workloads, deleting data, and removing k3s – so you can stop at any point. See Uninstall AxoConsole.

Advanced flags:

- `--skip-images` : Skip container image import entirely. Use when images are already loaded into k3s.
- `--image-mode bundle|component|skip` : Control how container images are imported. `bundle` (default) imports pre-packaged image tarballs; `component` imports one image per component; `skip` skips import.
- `--prepare-env` : Install k3s, cert-manager, and Traefik only – skip AxoConsole itself. Use for a two-phase install where you want to configure the environment before deploying AxoConsole.
- `--ignore-requirements` : Bypass hardware requirement checks.

4. If needed, follow the on-screen instructions. For a test installation, you can usually use the default values. For recommendations for a production environment, contact our support team.

5. Wait a few minutes until everything is installed.

   In case of any errors, check Troubleshooting for tips.

6. The script prompts to display the password of the default `admin` user. Record it, you'll need it to log in to AxoConsole.

```
...
[2026-06-02 20:44:28] [INFO ] All pods in axoflow are ready/complete
[2026-06-02 20:44:28] [INFO ] Axoflow is healthy
[2026-06-02 20:44:28] [INFO ] Access the UI at: https://<axoconsole-domain-
name>
Show admin credentials? [y/N]: y
Username: admin
Password: MzGIjuWKjXOtiRs4
```

   If needed, later you can retrieve the password by running:

```
kubectl -n "${AXOFLOW_NAMESPACE}" get secret axoidp-admin \
-o go-template='Username: {{ .data.username | base64decode }}{{ printf "\n"
}}Password: {{ .data.password | base64decode }}{{ printf "\n" }}'
```

> **Note**
>
> **When using AxoRouter with an on-premises AxoConsole deployment, you have to prepare the hosts before deploying AxoRouter. These steps are specific to on-premises AxoConsole deployments, and are not needed when using the SaaS AxoConsole.**

Login to AxoConsole

1. If the domain name of AxoConsole cannot be resolved from your desktop, add it to the `/etc/hosts` file in the following format. Use and IP address of AxoConsole that can be accessed from your desktop.

   ```
   <AXOFLOW-CONSOLE-IP-ADDRESS> <your-host.your-domain> idp.<your-host.your-
   domain> kcp.<your-host.your-domain> telemetry.<your-host.your-domain>
   ```

2. Open the `https://<your-host.your-domain>` URL in your browser.
3. The on-premise deployment of AxoConsole shows a self-signed certificate. If your browser complains about the related risks, accept it.
4. Use the email address and password you got or set in the installation step to log in to AxoConsole.

Prepare AxoRouter hosts

> **Note**
>
> **When using AxoRouter with an on-premises AxoConsole deployment, you have to prepare the hosts before deploying AxoRouter. These steps are specific to on-premises AxoConsole deployments, and are not needed when using the SaaS AxoConsole.**

AxoConsole service reference

Service Name	Namespace	Purposes	Function
KCP	`axoflow`	Backend API	Kubernetes Like Service with built in database, that service manage all the settings that our system manage
Chalco	`axoflow`	Frontend API	Serve the UI API Calls, implement business logic for the UI
Controller-Manager	`axoflow`	Backend Service	Reacts to state changes in our business entities, manage business logic for the Backend
Telemetry Proxy	`axoflow`	Backend API	Receives agents telemetries
UI	`axoflow`	Dashboard	The frontend for AxoConsole
Prometheus	`axoflow`	Backend API /Service	Monitoring component to store time series information and an API for query, manage alert rules
Alertmanager	`axoflow`	Backend API /Service	Monitoring component to send alerts based on alerting rules (optional, disabled by default)
Axoflow IDP	`axoflow`	Identity Connector/Proxy	Identity Connector/Proxy to allow the customer to use own identity (Google, LDAP, etc.)
Axolet Dist	`axoflow`	Backend API	Static artifact store to contains agents binaries
Cert Manager (kcp)	`axoflow`	Automated Certificate management tool	Manage certificates for Agents
Cert Manager	`cert-manager`	Automated Certificate management tool	Manage certificates for Axoflow components (Backend API, HTTP Proxy)
FilterX Validator	`axoflow`	Backend API	Validates FilterX expressions used in data processing pipelines
Traefik Ingress Controller	`ingress-traefik`	HTTP Proxy	Route the HTTP traffic between multiple Frontend/Backend APIs

Troubleshooting

- If you get the `[ERROR] Timed out waiting for k3s node to be Ready` error message, increase the timeout limit by running `K3S_TIMEOUT=180`, then rerun the installation script.

In case of other errors, run the following command to create a debug package from the cluster, and contact our support team.

```
sudo k3s kubectl cluster-info dump --all-namespaces  --output-directory=./cluster-dump && tar -czvf cluster-dump.tar.gz cluster-dump
```

Uninstall AxoConsole

To uninstall AxoConsole, run the installer with the `--uninstall` flag:

```
sudo ./axoflow-install.sh --uninstall
```

The uninstaller prompts before each step and requires separate confirmation for:

1. Removing the AxoConsole workloads and manifests
2. Deleting stored data (namespace and PersistentVolumeClaims) — **irreversible**

   > **CAUTION:**
   >
   > Hazard of data loss!
   >
   > Confirming step 2 permanently deletes all stored data.

3. Removing k3s

   > **CAUTION:**
   >
   > Hazard of data loss!
   >
   > If you're using k3s to run other software apart from AxoConsole, decline step 3 to keep k3s in place.

The uninstaller is safe to run on a partial or incomplete installation.

17.1.2 Authentication

Choose between the built-in identity provider and Dex for logging in to an on-premises AxoConsole.

AxoConsole supports the following authentication providers for on-premises deployments:

Provider	Description	When to use
Built-in (axoidp)	Axoflow's built-in identity provider	Default for all installations. Manages local users with username and password.
Dex	Open-source identity federation layer	Use when you want to authenticate users with an existing identity system such as LDAP, Google, or GitHub.

The built-in provider is enabled automatically during installation. To use a different provider, see the relevant section below.

17.1.2.1 Dex

Authenticate on-premises AxoConsole users with an existing identity system through Dex, and switch over from the built-in provider.

Dex is an open-source identity federation layer. When you configure AxoConsole to use Dex, Dex handles authentication with your existing identity system and passes user identity back to AxoConsole.

The following connectors are supported:

- LDAP – connect to an LDAP or Active Directory server
- GitHub – authenticate with GitHub OAuth
- Google – authenticate with Google accounts or Google Groups

Switch from the built-in provider to Dex

Configure Dex as the OIDC provider in the `/var/lib/rancher/k3s/server/manifests/axoflow-config.yaml` file. For example:

```
chalco:
  oidc:
    enabled: true
    provider: "dex"

axoidp:
  enabled: false

dex:
  enabled: true
```

After saving the file, k3s automatically applies the changes. Users will be redirected to the Dex login page on next login.

17.1.2.1.1 GitHub

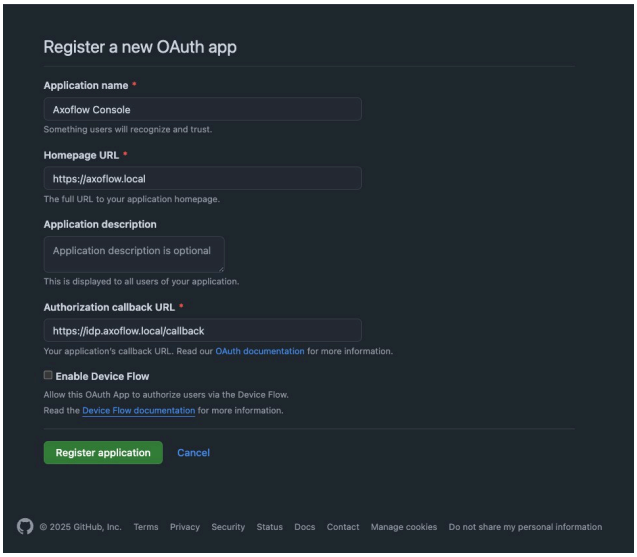
Log in to an on-premises AxoConsole with the GitHub accounts of your organization, using a GitHub OAuth application and Dex.

This section shows you how to use GitHub OAuth2 as an authentication backend for AxoConsole. It is assumed that you already have a GitHub organization. Complete the following steps.

Prerequisites

Register a new OAuth GitHub application for your organization. (For testing, you can create it under your personal account.) Make sure to:

- Set the **Homepage URL** to the URL of your AxoConsole deployment: `https://<your-console-host.your-domain>/callback`, for example, `https://axoflow-console.example.com`.
- Set the **Authorization callback URL** to: `https://auth.<your-console-host.your-domain>/callback`, for example, `https://auth.axoflow-console.example.com/callback`.



- Save the **Client id** of the app, you'll need it to configure AxoConsole.
- Generate a **Client secret** and save it, you'll need it to configure AxoConsole.

Configuration

1. Configure Dex as the OIDC provider in the `/var/lib/rancher/k3s/server/manifests/axoflow-config.yaml` file. For example:

```
chalco:
  oidc:
    enabled: true
    provider: "dex"

axoidp:
  enabled: false

dex:
  enabled: true
```

2. Configure authentication by editing the `spec.dex.config` section of the `/var/lib/rancher/k3s/server/manifests/axoflow-config.yaml` file.

    1. (Optional) If you've used our earlier example, delete the `spec.dex.config.staticPasswords` section.

    2. Add the `spec.dex.config.connectors` section to the file, like this:

```
dex:
  enabled: true
  localIP: $VM_IP_ADDRESS
  config:
    create: true
    connectors:
    - type: github
      # Required field for connector id.
      id: github
      # Required field for connector name.
      name: GitHub
      config:
        # Credentials can be string literals or pulled from the
environment.
        clientID: <ID-of-GitHub-application>
        clientSecret: <Secret-of-GitHub-application>
        redirectURI: <idp.your-host.your-domain/callback>
        orgs:
          - name: <your-GitHub-organization>
```

Note that if you have a valid domain name that points to the VM, you can omit the `localIP: $VM_IP_ADDRESS` line.

3. Edit the following fields. For details on the configuration parameters, see the Dex GitHub connector documentation.

   - `connectors.config.clientID` : The ID of the GitHub OAuth application.
   - `connectors.config.clientSecret` : The client secret of the GitHub OAuth application.
   - `connectors.config.redirectURI` : The callback URL of the GitHub OAuth application: `https://auth.<your-console-host.your-domain>/callback` , for example, `https://auth.axoflow-console.example.com/callback` .
   - If you want to fetch every GitHub group, replace the entire `orgs` section with the `loadAllGroups: true`
   - `connectors.config.orgs` : List of the GitHub organizations whose members can access AxoConsole. Restrict access to your organization.

3. Configure authorization in the `kcp.rbac.roles` section of the `/var/lib/rancher/k3s/server/manifests/axoflow-config.yaml` file.

   The following roles are available in AxoConsole by default:

   - `IAMadmin` : Can manage the roles and permissions to access AxoConsole. Only this role can make changes on the **Settings > Roles** page. If your identity provider has an administration page, this role can also see the **Identity Provider Admin Panel** link to it on the **Settings > Roles** page. (If your identity provider has a profile page, every user can open their own profile by selecting **<their username> > Profile** in the top-right corner.)

   - `admin` : Has full access to AxoConsole, except:

     - has only read access to the **Settings > Roles** page,
     - can't search logs in stores (that requires the `logSearcher` role, or access role to a specific store).

   - `infrastructureManager` : Can manage the infrastructure (without the permissions to view or tap log contents, or to rehydrate data). Has full access to the following pages: **Activity Logs**, **Alerting**, **Routers**, **Sources**, **Flows**, **Provisioning**, **Search Logs**. Can tap into service logs which may contain event payload.

   - `infrastructureViewer` : Similar to **infrastructure-manager**, but can only view the pages.

   - `contentManager` : Can view log content and manage content related details like flows (without the permissions to manage infrastructure, but including access to view infrastructure details).

    - Has full access to the **Rehydration** page.
    - Can view and modify **Flows**, but can't create or delete them.
    - Can view **Routers**, **Sources**, **Search Logs**, **Analytics**.
    - Can tap into logs.
- `contentViewer`: Can view content like analytics, log search, log tapping (without the permissions to manage or view infrastructure details).
    - Can view **Search Logs**, **Analytics**.
    - Can tap into logs.
- `logSearcher`: Can search in every store. You can also grant access to specific stores.

If you need other roles, contact the Axoflow support team. Composing other roles is possible as part of a custom integration.

You can use email addresses or GitHub groups to assign your users to AxoConsole user roles.

To use the groups fetched from GitHub, prefix the name of the GitHub group with `groups:`, and add them to the related roles under `kcp.rbac.roles.<role>.groups`. For example:

```
rbac:
  createRoles: true
  createRoleBindings: true
  roles:
    IAMadmin:
      groups:
        - groups:iamadmin
    admin:
      groups:
        - groups:admins
    contentViewer:
      groups:
        - groups:readonly
```

- A specific email address, for example, `email:username@example.com`

> **Note**
>
> - The user must be able to authenticate to AxoConsole using this email address, so authentication must be properly set up.
> - Multiple identity providers may allow authentication with the same email address, possibly with different level of ownership validation.

- An entire email domain, for example, `emaildomain:example.com`. Any user who authenticates with an email address belonging to this email domain will have access to the role.
- A user group. The format of the group depends on the OID provider.
  - If you're using AxoConsole as a SaaS, specify the group in the following format: `cognito:groups:<groupname-retrieved-from-cognito>`, for example, `cognito:groups:ExampleGoogleSaml:Operator`.
  - If you're using an on-prem AxoConsole deployment, specify the group in the following format: `groups:<groupname-retrieved-from-dex>` for example, `groups:operator`.
- A specific session of an authenticated user, in the following format: `user:<oidc-response-subject>`, for example: `user:42c4e962-f077-705f-138f-f01ba1220c44`.
- Every authenticated user: `axoflow:user`

For details on authorization settings, see Authorization.

4. Save the file.
5. Restart the dex deployment after changing the connector:

```
kubectl rollout restart deployment/dex -n axoflow
```

Expected output:

```
deployment.apps/dex restarted
```

6. Open the main page of your AxoConsole deployment in your browser. You'll be redirected to the GitHub authentication page.

   After completing the GitHub authentication you can access AxoConsole.

Getting help

You can troubleshoot common errors by running `kubectl logs -n axoflow <dex-container-name>`

If you run into problems setting up the authentication or authorization, contact our support team.

17.1.2.1.2 Google

Log in to an on-premises AxoConsole with Google accounts, using Google OpenID Connect and Dex.

This section shows you how to use Google OpenID Connect as an authentication backend for AxoConsole. It is assumed that you already have a Google organization and Google Cloud Console access. Complete the following steps.

Prerequisites

- To use Google authentication, AxoConsole must be deployed on a publicly accessible domain name (the `$BASE_HOSTNAME` must end with a valid top-level domain, for example, `.com` or `.io`.)
- Configure OpenID Connect and create an OpenID credential for a **Web application**.
  - Make sure to set the **Authorized redirect URIs** to: `https://auth.<your-console-host.your-domain>/callback`, for example, `https://auth.axoflow-console.example.com/callback`.
  - Save the **Client ID** of the app and the **Client secret** of the application, you'll need them to configure AxoConsole.

  For details on setting up OpenID Connect, see the official documentation.

Google group membership

To use group memberships when assigning your users to AxoConsole user roles, you have to configure a service account to retrieve group memberships. Otherwise, you can use email domains. To set up a proper service account, complete the following steps.

1. Set up a service account with Domain-Wide Delegation.
2. Save the JSON key file that contains the authentication information for the service account. You'll need it later.
3. Delegate the `https://www.googleapis.com/auth/admin.directory.group.readonly` scope to the service account. Do not delegate any other scope.
4. Enable the Admin SDK.

You'll also need the email address of a Google Workspace user with a minimum of the Groups Reader Role assigned. The service account will impersonate this user when making calls to the admin API.

Configuration

1. Configure Dex as the OIDC provider in the `/var/lib/rancher/k3s/server/manifests/axoflow-config.yaml` file. For example:

```
chalco:
  oidc:
    enabled: true
    provider: "dex"

axoidp:
  enabled: false

dex:
  enabled: true
```

2. Configure authentication by editing the `spec.dex.config` section of the `/var/lib/rancher/k3s/server/manifests/axoflow-config.yaml` file.

    1. (Optional) If you've used our earlier example, delete the `spec.dex.config.staticPasswords` section.

    2. Add the `spec.dex.config.connectors` section to the file, like this:

```
connectors:
- type: google
  id: google
  name: Google
  config:
    # Connector config values starting with a "$" will read from the
environment.
    clientID: <ID-of-Google-application>
    clientSecret: <Secret-of-GitHub-application>

    # Dex's issuer URL + "/callback"
    redirectURI: <idp.your-host.your-domain/callback>

    # Set the value of `prompt` query parameter in the authorization
request
    # The default value is "consent" when not set.
    promptType: consent

    # Google supports whitelisting allowed domains when using G Suite
    # (Google Apps). The following field can be set to a list of domains
    # that can log in:
    #
    hostedDomains:
    - <your-domain>

    # For group membership authorization:
    #serviceAccountFilePath: "/etc/dex/sa-file/service-account.json"
    #domainToAdminEmail:
      #"*": <impersonated-group-reader-email-address>
```

3. Edit the following fields. For details on the configuration parameters, see the Dex Google connector documentation.

   - `connectors.config.clientID` : The ID of the Google application.
   - `connectors.config.clientSecret` : The client secret of the Google application.
   - `connectors.config.redirectURI` : The callback URL of the Google application: `https://auth.<your-console-host.your-domain>/callback` , for example, `https://auth.axoflow-console.example.com/callback` .

- `connectors.config.hostedDomains`: The domain where AxoConsole is deployed, for example, `example.com`. Your users must have email addresses for this domain at Google.

4. If you'll use Google group memberships for authorization, uncomment the `serviceAccountFilePath` and `domainToAdminEmail` sections. Replace `<impersonated-group-reader-email-address>` with the email address you created in Google group memberships prerequisites.

```
# For group membership authorization:
serviceAccountFilePath: "/etc/dex/sa-file/service-account.json"
domainToAdminEmail:
  "*": <impersonated-group-reader-email-address>
```

3. Configure authorization in the `kcp.rbac.roles` section of the `/var/lib/rancher/k3s/server/manifests/axoflow-config.yaml` file.

   The following roles are available in AxoConsole by default:

   - `IAMadmin`: Can manage the roles and permissions to access AxoConsole. Only this role can make changes on the **Settings > Roles** page. If your identity provider has an administration page, this role can also see the **Identity Provider Admin Panel** link to it on the **Settings > Roles** page. (If your identity provider has a profile page, every user can open their own profile by selecting **<their username> > Profile** in the top-right corner.)
   - `admin`: Has full access to AxoConsole, except:
     - has only read access to the **Settings > Roles** page,
     - can't search logs in stores (that requires the `logSearcher` role, or access role to a specific store).
   - `infrastructureManager`: Can manage the infrastructure (without the permissions to view or tap log contents, or to rehydrate data). Has full access to the following pages: **Activity Logs**, **Alerting**, **Routers**, **Sources**, **Flows**, **Provisioning**, **Search Logs**. Can tap into service logs which may contain event payload.
   - `infrastructureViewer`: Similar to **infrastructure-manager**, but can only view the pages.
   - `contentManager`: Can view log content and manage content related details like flows (without the permissions to manage infrastructure, but including access to view infrastructure details).
     - Has full access to the **Rehydration** page.

    - Can view and modify **Flows**, but can't create or delete them.
    - Can view **Routers**, **Sources**, **Search Logs**, **Analytics**.
    - Can tap into logs.
- `contentViewer`: Can view content like analytics, log search, log tapping (without the permissions to manage or view infrastructure details).
    - Can view **Search Logs**, **Analytics**.
    - Can tap into logs.
- `logSearcher`: Can search in every store. You can also grant access to specific stores.

If you need other roles, contact the Axoflow support team. Composing other roles is possible as part of a custom integration.

To assign Google groups to AxoConsole roles, add the groups of your users to the related roles under `kcp.rbac.roles.<role>.groups`.

> **Note**
>
> **Make sure that you've completed the prerequisites detailed in Google group membership.**

```
rbac:
  createRoles: true
  createRoleBindings: true
  roles:
    IAMadmin:
      groups:
        - groups:iamadmin@example.com
    admin:
      groups:
        - groups:admin@example.com
    contentViewer:
      groups:
        - groups:readonly@example.com
```

- A specific email address, for example, `email:username@example.com`

> **Note**
>
> - The user must be able to authenticate to AxoConsole using this email address, so authentication must be properly set up.
> - Multiple identity providers may allow authentication with the same email address, possibly with different level of ownership validation.

- An entire email domain, for example, `emaildomain:example.com`. Any user who authenticates with an email address belonging to this email domain will have access to the role.
- A user group. The format of the group depends on the OID provider.
  - If you're using AxoConsole as a SaaS, specify the group in the following format: `cognito:groups:<groupname-retrieved-from-cognito>`, for example, `cognito:groups:ExampleGoogleSaml:Operator`.
  - If you're using an on-prem AxoConsole deployment, specify the group in the following format: `groups:<groupname-retrieved-from-dex>` for example, `groups:operator`.
- A specific session of an authenticated user, in the following format: `user:<oidc-response-subject>`, for example: `user:42c4e962-f077-705f-138f-f01ba1220c44`.
- Every authenticated user: `axoflow:user`

For details on authorization settings, see Authorization.

4. Save the file.
5. If you're using Google groups to assign user roles, complete the following steps.
   1. Run the following command to delete the `dex-google-service-account` secret. This is a placeholder secret that you'll re-create with valid values in the next step.

      ```
      kubectl -n axoflow delete secret dex-google-service-account
      ```

   2. Re-create the secret using the JSON key of the Google service account you've created in Google group membership.

      ```
      kubectl -n axoflow create secret dex-google-service-account --from-file
      <path-to-the-JSON-key>
      ```

      Expected output:

```
secret/dex-google-service-account created
```

3. Edit the `/var/lib/rancher/k3s/server/manifests/axoflow-config.yaml` file and set the `dex.googleServiceAccount.create` option to `false`.

6. Restart the dex deployment after changing the connector:

```
kubectl rollout restart deployment/dex -n axoflow
```

Expected output:

```
deployment.apps/dex restarted
```

7. Open the main page of your AxoConsole deployment in your browser. You'll be redirected to the Google authentication page.

   After completing the Google authentication you can access AxoConsole.

Getting help

You can troubleshoot common errors by running `kubectl logs -n axoflow <dex-container-name>`

If you run into problems setting up the authentication or authorization, contact our support team.

17.1.2.1.3 LDAP

Log in to an on-premises AxoConsole with the accounts of your LDAP server, using Dex.

This section shows you how to use LDAP as an authentication backend for AxoConsole. In the examples we used the public demo service of FreeIPA as an LDAP server. It is assumed that you already have an LDAP server in place. Complete the following steps.

1. Configure Dex as the OIDC provider in the `/var/lib/rancher/k3s/server/manifests/axoflow-config.yaml` file. For example:

```
chalco:
  oidc:
    enabled: true
    provider: "dex"

axoidp:
  enabled: false

dex:
  enabled: true
```

2. Configure authentication by editing the `spec.dex.config` section of the `/var/lib/rancher/k3s/server/manifests/axoflow-config.yaml` file.
    1. (Optional) If you've used our earlier example, delete the `spec.dex.config.staticPasswords` section.
    2. Add the `spec.dex.config.connectors` section to the file, like this:

    > **CAUTION:**
    >
    > This example shows a simple configuration suitable for testing. In production environments, make sure to:
    >
    > - configure TLS encryption to access your LDAP server
    > - retrieve the bind password from a vault or environment variable. Note that if the bind password contains the `$` character, you must set it in an environment variable and pass it like `bindPW: $LDAP_BINDPW`.

```
    dex:
      enabled: true
      localIP: $VM_IP_ADDRESS
      config:
        create: true
        connectors:
        - type: ldap
          name: OpenLDAP
          id: ldap
          config:
            host: ipa.demo1.freeipa.org
            insecureNoSSL: true
            # This would normally be a read-only user.
            bindDN:
uid=admin,cn=users,cn=accounts,dc=demo1,dc=freeipa,dc=org
            bindPW: Secret123
            usernamePrompt: Email Address
            userSearch:
              baseDN: dc=demo1,dc=freeipa,dc=org
              filter: "(objectClass=person)"
              username: mail
              # "DN" (case sensitive) is a special attribute name. It
indicates that
              # this value should be taken from the entity's DN not an
attribute on
              # the entity.
              idAttr: uid
              emailAttr: mail
              nameAttr: cn
            groupSearch:
              baseDN: dc=demo1,dc=freeipa,dc=org
              filter: "(objectClass=groupOfNames)"
              userMatchers:
                # A user is a member of a group when their DN matches
                # the value of a "member" attribute on the group entity.
              - userAttr: DN
                groupAttr: member
              # The group name should be the "cn" value.
              nameAttr: cn
```

3. Edit the following fields. For details on the configuration parameters, see the Dex LDAP connector documentation.

   - `connectors.config.host` : The hostname and optionally the port of the LDAP server in “host:port” format.
   - `connectors.config.bindDN` and `connectors.config.bindPW` : The DN and password for an application service account that the connector uses to search for users and groups.
   - `connectors.config.userSearch.bindDN` and `connectors.config.groupSearch.bindDN` : The base DN for the user and group search.

3. Configure authorization in the `kcp.rbac.roles` section of the `/var/lib/rancher/k3s/server/manifests/axoflow-config.yaml` file.

   The following roles are available in AxoConsole by default:

   - `IAMadmin` : Can manage the roles and permissions to access AxoConsole. Only this role can make changes on the **Settings > Roles** page. If your identity provider has an administration page, this role can also see the **Identity Provider Admin Panel** link to it on the **Settings > Roles** page. (If your identity provider has a profile page, every user can open their own profile by selecting **<their username> > Profile** in the top-right corner.)
   - `admin` : Has full access to AxoConsole, except:
     - has only read access to the **Settings > Roles** page,
     - can’t search logs in stores (that requires the `logSearcher` role, or access role to a specific store).
   - `infrastructureManager` : Can manage the infrastructure (without the permissions to view or tap log contents, or to rehydrate data). Has full access to the following pages: **Activity Logs**, **Alerting**, **Routers**, **Sources**, **Flows**, **Provisioning**, **Search Logs**. Can tap into service logs which may contain event payload.
   - `infrastructureViewer` : Similar to **infrastructure-manager**, but can only view the pages.
   - `contentManager` : Can view log content and manage content related details like flows (without the permissions to manage infrastructure, but including access to view infrastructure details).
     - Has full access to the **Rehydration** page.
     - Can view and modify **Flows**, but can’t create or delete them.
     - Can view **Routers**, **Sources**, **Search Logs**, **Analytics**.
     - Can tap into logs.

- `contentViewer`: Can view content like analytics, log search, log tapping (without the permissions to manage or view infrastructure details).
  - Can view **Search Logs**, **Analytics**.
  - Can tap into logs.
- `logSearcher`: Can search in every store. You can also grant access to specific stores.

If you need other roles, contact the Axoflow support team. Composing other roles is possible as part of a custom integration.

- Add the names of the LDAP groups to the related roles under `kcp.rbac.roles.<role>.groups`. In the following example, the `managers` group gets administrator role, and the `readonly` group gets read-only access to AxoConsole.

```
rbac:
  createRoles: true
  createRoleBindings: true
  roles:
    IAMadmin:
      groups:
        - groups:iamadmin
    admin:
      groups:
        - groups:managers
    contentViewer:
      groups:
        - groups:readonly
```

- A specific email address, for example, `email:username@example.com`

> **Note**
>
> - The user must be able to authenticate to AxoConsole using this email address, so authentication must be properly set up.
> - Multiple identity providers may allow authentication with the same email address, possibly with different level of ownership validation.

- An entire email domain, for example, `emaildomain:example.com`. Any user who authenticates with an email address belonging to this email domain will have access to the role.

- A user group. The format of the group depends on the OID provider.
  - If you're using AxoConsole as a SaaS, specify the group in the following format: `cognito:groups:<groupname-retrieved-from-cognito>`, for example, `cognito:groups:ExampleGoogleSaml:Operator`.
  - If you're using an on-prem AxoConsole deployment, specify the group in the following format: `groups:<groupname-retrieved-from-dex>` for example, `groups:operator`.
- A specific session of an authenticated user, in the following format: `user:<oidc-response-subject>`, for example: `user:42c4e962-f077-705f-138f-f01ba1220c44`.
- Every authenticated user: `axoflow:user`

For details on authorization settings, see Authorization.

4. Save the file.
5. Restart the dex deployment after changing the connector:

```
kubectl rollout restart deployment/dex -n axoflow
```

Expected output:

```
deployment.apps/dex restarted
```

Getting help

You can troubleshoot common errors by running `kubectl logs -n axoflow <dex-container-name>`

If you run into problems setting up the authentication or authorization, contact our support team.

17.1.3 Authorization

Look up the roles of an on-premises AxoConsole, and bind them to the users and groups of your authentication backend.

These sections show you how to configure the authorization of AxoConsole with different authentication backends.

Authorization is configured in the `kcp.rbac.roles` section of `/var/lib/rancher/k3s/server/manifests/axoflow-config.yaml`.

Default configuration

The installer sets up the following default role bindings, which grant full access to users in the `admin` group of your identity provider:

```
kcp:
  rbac:
    roles:
      IAMadmin:
        groups:
          - "groups:admin"
      admin:
        groups:
          - "groups:admin"
```

Verify that this matches your requirements by checking the file:

```
grep -A20 'rbac:' /var/lib/rancher/k3s/server/manifests/axoflow-config.yaml
```

Available roles

The following roles are available in AxoConsole by default:

- `IAMadmin`: Can manage the roles and permissions to access AxoConsole. Only this role can make changes on the **Settings > Roles** page. If your identity provider has an administration page, this role can also see the **Identity Provider Admin Panel** link to it

on the **Settings > Roles** page. (If your identity provider has a profile page, every user can open their own profile by selecting **<their username> > Profile** in the top-right corner.)

- `admin` : Has full access to AxoConsole, except:
  - has only read access to the **Settings > Roles** page,
  - can't search logs in stores (that requires the `logSearcher` role, or access role to a specific store).
- `infrastructureManager` : Can manage the infrastructure (without the permissions to view or tap log contents, or to rehydrate data). Has full access to the following pages: **Activity Logs**, **Alerting**, **Routers**, **Sources**, **Flows**, **Provisioning**, **Search Logs**. Can tap into service logs which may contain event payload.
- `infrastructureViewer` : Similar to **infrastructure-manager**, but can only view the pages.
- `contentManager` : Can view log content and manage content related details like flows (without the permissions to manage infrastructure, but including access to view infrastructure details).
  - Has full access to the **Rehydration** page.
  - Can view and modify **Flows**, but can't create or delete them.
  - Can view **Routers**, **Sources**, **Search Logs**, **Analytics**.
  - Can tap into logs.
- `contentViewer` : Can view content like analytics, log search, log tapping (without the permissions to manage or view infrastructure details).
  - Can view **Search Logs**, **Analytics**.
  - Can tap into logs.
- `logSearcher` : Can search in every store. You can also grant access to specific stores.

If you need other roles, contact the Axoflow support team. Composing other roles is possible as part of a custom integration.

Customize role bindings

To assign additional groups to a role, add them to the `groups` list. The group format depends on your authentication backend:

- Built-in identity provider (axoidp) and LDAP: `groups:<group-name>`
- Email domain: `emaildomain:<domain>`

Edit the manifest and save – k3s picks up the change automatically:

```
sudo vi /var/lib/rancher/k3s/server/manifests/axoflow-config.yaml
```

17.1.4 Prepare AxoRouter hosts

Prepare hosts to run AxoRouter with an on-premises AxoConsole, with the generated script or manually.

When using AxoRouter with an on-premises AxoConsole deployment, you have to complete the following steps on the hosts you want to deploy AxoRouter on. These steps are specific to on-premises AxoConsole deployments, and are not needed when using the SaaS AxoConsole.

Using the generated prepare-host script

After a successful AxoConsole installation, the installer writes a `prepare-axorouter-host.sh` script to the log directory on the AxoConsole host. This script automates all the steps below.

1. Copy the script to each AxoRouter or Axoflow agent host.
2. Run it as root:

   ```
   sudo bash prepare-axorouter-host.sh
   ```

   The script adds the AxoConsole base hostname and its `idp.`, `kcp.`, and `telemetry.` subdomains to `/etc/hosts`, and installs the AxoConsole CA certificate into the system trust store.

Manual preparation

If the prepare-host script is not available, complete the following steps manually.

1. If the domain name of AxoConsole cannot be resolved from the AxoRouter host, add it to the `/etc/hosts` file of the AxoRouter host in the following format. Use and IP address of AxoConsole that can be accessed from the AxoRouter host.

   ```
   <AXOFLOW-CONSOLE-IP-ADDRESS> <AXOFLOW-CONSOLE-BASE-URL> idp.<AXOFLOW-CONSOLE-BASE-URL> kcp.<AXOFLOW-CONSOLE-BASE-URL> telemetry.<AXOFLOW-CONSOLE-BASE-URL>
   ```

2. Import AxoConsole certificates to AxoRouter hosts.

1. **On the AxoConsole host**: Run the following command to extract the AxoConsole CA certificate. The AxoRouter host will need this certificate to download the installation binaries.

   ```
   kubectl get secret -n axoflow axoflow-local-root-ca -o
   jsonpath='{.data.ca\.crt}' | base64 -d  > axoflow-ca.crt
   ```

   Copy this file to the AxoRouter hosts.

2. **On the AxoRouter hosts**: Copy the certificate file extracted from the AxoConsole host.

   - **On Red Hat**: Copy the files into the `/etc/pki/ca-trust/source/anchors/` folder, then run `sudo update-ca-trust extract`. (If needed, install the `ca-certificates` package.)
   - **On Ubuntu:** Copy the files into the `/usr/local/share/ca-certificates/` folder, then run `sudo update-ca-certificates`

3. `curl -I https://<your-host.your-domain>` should give you a valid HTTP/2 200 response

4. Now you can deploy AxoRouter on the host.

> ## Note
>
> **If you want to download the AxoRouter images directly from AxoConsole (useful in air-gapped environments), disable the External image pull option of the provisioning script.**

17.1.5 Upgrade

Upgrade an on-premises AxoConsole to a new version, then upgrade your AxoRouter and edge hosts.

To upgrade an on-prem AxoConsole deployment, complete the following steps.

1. Log in to the host running AxoConsole.
2. Run the following command to update the version number in the `/var/lib/rancher/k3s/server/manifests/axoflow-config.yaml` file to 0.90.0.

   ```
   sudo sed -i.old -E 's/^( *version: ).*/\10.90.0/'
   /var/lib/rancher/k3s/server/manifests/axoflow-config.yaml
   ```

3. Wait a few minutes until the pods are restarted. You can check their status by running:

   ```
   kubectl get pods -A
   ```

4. If everything is working as expected, re-deploy your AxoRouter and Axoflow agent instances to upgrade them.

In case you experience problems during the upgrade, contact our support team.

17.2 Air-gapped

Install AxoConsole on a virtual machine without internet access, and prepare the AxoRouter hosts that connect to it.

This guide shows you how to install AxoConsole in an air-gapped environment on a virtual machine. To deploy an on-premises AxoConsole that has internet access, see Deployment guide.

Since this is a single instance deployment, we don't recommend using it in production environments. For additional steps and configurations needed in production environments, contact our support team.

At a high level, the deployment consists of the following steps:

1. Preparing a virtual machine.
2. Running the installation script on the virtual machine that deploys:
   - Kubernetes
   - cert-manager and Traefik
   - AxoConsole

   Any of these components can be skipped if already installed.
3. Basic authentication with admin user and password is configured by default. You can also configure other common authentication methods like LDAP, Github and Google.
4. Before deploying AxoRouter describes the steps you have to complete on a host before deploying AxoRouter on it. These steps are specific to on-premises AxoConsole deployments, and aren't needed when using the SaaS AxoConsole.

Prerequisites

To install AxoConsole, you'll need the following:

- The URL for downloading the AxoConsole bundle. You'll receive this URL from our team. You can request it using the contact form.

  > **CAUTION:**
  >
  > Don't start the Install AxoConsole process until you've received the bundle URL.

- A license key for AxoConsole. You'll receive this from our team. You can request it using the contact form.
- A host that meets the system requirements.

System requirements

Supported platforms:

- Ubuntu 24.04 and 26.04,
- Red Hat Enterprise Linux 9.4 or newer and 10.

Other compatible and derivative distributions (for example, AlmaLinux or Rocky Linux) should work as well, but we don't test them explicitly.

The virtual machine (VM) must have at least:

Resource	Minimum	Production
CPU	4 vCPU	16 vCPU
RAM	8 GB	16 GB
Disk (`/`)	100 GB	250 GB

- A Minimum setup with 4 vCPU (x86_64-based), 8 GB RAM, and 100 GB disk space can handle about 100 AxoRouter instances and 1000 data source hosts.
- A real-life production scenario that handles 100 AxoRouter and 3000 data source hosts with 30-day metric retention would need:
  - 16 vCPU (x86_64-based)
  - 16 GB RAM
  - 250 GB disk space

> ## Note
>
> **Note that AxoRouter and Axoflow agent collects detailed, real-time metrics about the data-flows – giving you observability over the health of the security data pipeline and its components. Your security data remains in your self-managed cloud or in your on-prem instance where your sources, destinations, Axoflow agents, and AxoRouters are running, only metrics are forwarded to AxoConsole.**

For details on sizing, contact our support team.

You'll need to have access to a user with `sudo` privileges.

Install AxoConsole

CAUTION:

This procedure covers how to deploy a new AxoConsole instance. The installation script can't upgrade existing deployments, contact the Axoflow support team for detailed upgrade instructions.

1. Download the installation bundle file from the URL you've received from the Axoflow Support Team.
2. Transfer the file to the host where you want to install AxoConsole.

   Extract the file and navigate into the directory.

```
tar -xf axoflow-all-in-one-0.90.0-linux-amd64.tar
cd axoflow-all-in-one-0.90.0-linux-amd64
```

3. Run the installation script with your license key and the domain name for your AxoConsole deployment.

> **Note**
>
> **Re-running the installer will reuse any existing or modified manifests or user settings (for example, authentication settings added to `/var/lib/rancher/k3s/server/manifests/axoflow-config.yaml`) available on the host.**

```
sudo LICENSE_KEY=<your-license-key> BASE_HOSTNAME=<axoconsole-domain-name>
AXOFLOW_VERSION=0.90.0 bash ./axoflow-install.sh
```

> **Note**
>
> **If you omit `BASE_HOSTNAME`, the installer probes whether `<VM_IP>.nip.io` resolves and offers it as the default. nip.io is a free wildcard DNS service – handy for testing and demos because the console and all its subdomains resolve without any DNS or `/etc/hosts` configuration. For production deployments, use a proper domain name.**

You can use the following flags as needed for your environment.

- `--step` : Prompt before each step

- `--dry-run` : Show what would be done without executing anything
- `--skip-k3s` : Skip k3s installation (already installed)
- `--skip-cert-manager` : Skip cert-manager installation
- `--disable-traefik` : Disable the built-in Traefik ingress controller of k3s. Use this option when using a custom ingress controller.
- `--skip-nip-io` : Skip the automatic `<VM_IP>.nip.io` DNS probe and use `axoflow.local` as the default base hostname.
- `--diagnose [namespace]` : Inspect pods that are not ready and explain why (image-pull errors, configuration issues, crashes). Runs automatically on a timeout; use on demand after a failed install.
- `--uninstall` : Uninstall AxoConsole. Prompts before each step – removing workloads, deleting data, and removing k3s – so you can stop at any point. See Uninstall AxoConsole.

Advanced flags:

- `--skip-images` : Skip container image import entirely. Use when images are already loaded into k3s.
- `--image-mode bundle|component|skip` : Control how container images are imported. `bundle` (default) imports pre-packaged image tarballs; `component` imports one image per component; `skip` skips import.
- `--prepare-env` : Install k3s, cert-manager, and Traefik only – skip AxoConsole itself. Use for a two-phase install where you want to configure the environment before deploying AxoConsole.
- `--ignore-requirements` : Bypass hardware requirement checks.

4. If needed, follow the on-screen instructions. For a test installation, you can usually use the default values. For recommendations for a production environment, contact our support team.

5. Wait a few minutes until everything is installed.

   In case of any errors, check Troubleshooting for tips.

6. The script prompts to display the password of the default `admin` user. Record it, you'll need it to log in to AxoConsole.

```
...
[2026-06-02 20:44:28] [INFO ] All pods in axoflow are ready/complete
[2026-06-02 20:44:28] [INFO ] Axoflow is healthy
[2026-06-02 20:44:28] [INFO ] Access the UI at: https://<axoconsole-domain-
name>
Show admin credentials? [y/N]: y
Username: admin
Password: MzGIjuWKjXOtiRs4
```

If needed, later you can retrieve the password by running:

```
kubectl -n "${AXOFLOW_NAMESPACE}" get secret axoidp-admin \
-o go-template='Username: {{ .data.username | base64decode }}{{ printf "\n"
}}Password: {{ .data.password | base64decode }}{{ printf "\n" }}'
```

> **Note**
>
> **When using AxoRouter with an on-premises AxoConsole deployment, you have to prepare the hosts before deploying AxoRouter. These steps are specific to on-premises AxoConsole deployments, and are not needed when using the SaaS AxoConsole.**

Login to AxoConsole

1. If the domain name of AxoConsole cannot be resolved from your desktop, add it to the `/etc/hosts` file in the following format. Use and IP address of AxoConsole that can be accessed from your desktop.

   ```
   <AXOFLOW-CONSOLE-IP-ADDRESS> <your-host.your-domain> idp.<your-host.your-
   domain> kcp.<your-host.your-domain> telemetry.<your-host.your-domain>
   ```

2. Open the `https://<your-host.your-domain>` URL in your browser.

3. The on-premise deployment of AxoConsole shows a self-signed certificate. If your browser complains about the related risks, accept it.

4. Use the email address and password you got or set in the installation step to log in to AxoConsole.

Firewall configuration

If you run a host firewall (for example, `firewalld`), open the ports AxoConsole and AxoRouter need before you start the installation, otherwise agents and log sources won't be able to connect.

> ### OS- and runtime-specific
>
> The reference commands in this section are written for **RHEL 9.X** and the **k3s** Kubernetes distribution, which is the default in the air-gapped AxoConsole bundle. If you run a different operating system, a different firewall front-end, or a Kubernetes distribution other than k3s, the commands, zones, and CIDRs may differ: use the port list and adapt the commands to your environment. For the operating system requirements of k3s, see the k3s installation requirements.
>
> The k3s project recommends disabling firewalld entirely (`systemctl disable --now firewalld`) and relying on k3s-managed iptables rules instead. Only apply the rules below if your security policy requires `firewalld` to stay enabled.

Ports to open

On the AxoConsole node:

The AxoConsole host must have the following **Open Ports**:

- 80 TCP: Traefik ingress (HTTP, redirected to HTTPS).
- 443 TCP: HTTPS for the AxoConsole UI, API, and all web traffic.
- 6443 TCP: k3s API server, required for `kubectl` and for the `axolet` agents on AxoRouter nodes to connect back to the cluster.

In addition, the k3s pod and service CIDRs (by default `10.42.0.0/16` and `10.43.0.0/16`) must be trusted so that intra-cluster traffic is not filtered.

If you deploy AxoRouter on the AxoConsole host, open the ports needed by AxoRouter. For details, see Install AxoRouter on Linux.

Reference rules for RHEL and k3s (default in air-gapped setup)

Show the reference `firewalld` commands for RHEL 9.X and k3s.

> **Note**
>
> **These commands assume RHEL 9.X with `firewalld` and a default k3s installation (with Traefik as the ingress controller). Adjust the pod and service CIDRs only if you've changed the k3s defaults (`--cluster-cidr` / `--service-cidr`).**

On the AxoConsole node:

```
# Traefik ingress - HTTP redirect and HTTPS (Console UI, API, all web traffic)
firewall-cmd --permanent --add-port=80/tcp
firewall-cmd --permanent --add-port=443/tcp

# k3s API server - required for kubectl and for axolet agents on AxoRouter nodes
firewall-cmd --permanent --add-port=6443/tcp

# Trust k3s pod and service CIDRs so intra-cluster traffic is not filtered
# Adjust these only if you changed the k3s defaults (--cluster-cidr / --service-
cidr)
firewall-cmd --permanent --zone=trusted --add-source=10.42.0.0/16  # pod CIDR
firewall-cmd --permanent --zone=trusted --add-source=10.43.0.0/16  # service
CIDR

firewall-cmd --reload
```

If you've deployed AxoRouter on the node, open the log-ingestion ports that correspond to the default `portMapping` in the `axorouter-syslog` chart. Disable or skip any port your deployment does not use.

```
# Syslog - standard UDP and TCP
firewall-cmd --permanent --add-port=514/udp
firewall-cmd --permanent --add-port=514/tcp

# Syslog - reliable delivery (RFC 3195)
firewall-cmd --permanent --add-port=601/tcp

# Syslog - TLS (RFC 5425)
firewall-cmd --permanent --add-port=6514/tcp

# OpenTelemetry - gRPC receiver
firewall-cmd --permanent --add-port=4317/tcp

# Elasticsearch Bulk API receiver (optional - only if you forward Elasticsearch
traffic)
firewall-cmd --permanent --add-port=9200/tcp

# Splunk HEC receiver (optional - only if you forward Splunk HEC traffic)
firewall-cmd --permanent --add-port=8088/tcp

# Windows Event Collector (WEC) LoadBalancer service (optional)
firewall-cmd --permanent --add-port=5085/tcp

firewall-cmd --reload
```

When you create a connector rule in AxoConsole that opens a new listening port on a AxoRouter node, open that port in `firewalld` on the corresponding node as well:

```
# Replace <PORT> and <PROTOCOL> (tcp or udp) with the values from your connector
rule
firewall-cmd --permanent --add-port=<PORT>/<PROTOCOL>
firewall-cmd --reload
```

To list all currently open ports:

```
firewall-cmd --list-ports
```

Prepare AxoRouter hosts

> **Note**
>
> **When using AxoRouter with an on-premises AxoConsole deployment, you have to prepare the hosts before deploying AxoRouter. These steps are specific to on-premises AxoConsole deployments, and are not needed when using the SaaS AxoConsole.**

AxoConsole service reference

Service Name	Namespace	Purposes	Function
KCP	`axoflow`	Backend API	Kubernetes Like Service with built in database, that service manage all the settings that our system manage
Chalco	`axoflow`	Frontend API	Serve the UI API Calls, implement business logic for the UI
Controller-Manager	`axoflow`	Backend Service	Reacts to state changes in our business entities, manage business logic for the Backend
Telemetry Proxy	`axoflow`	Backend API	Receives agents telemetries
UI	`axoflow`	Dashboard	The frontend for AxoConsole
Prometheus	`axoflow`	Backend API /Service	Monitoring component to store time series information and an API for query, manage alert rules
Alertmanager	`axoflow`	Backend API /Service	Monitoring component to send alerts based on alerting rules (optional, disabled by default)
Axoflow IDP	`axoflow`	Identity Connector/Proxy	Identity Connector/Proxy to allow the customer to use own identity (Google, LDAP, etc.)
Axolet Dist	`axoflow`	Backend API	Static artifact store to contains agents binaries
Cert Manager (kcp)	`axoflow`	Automated Certificate management tool	Manage certificates for Agents

Cert Manager	`cert-manager`	Automated Certificate management tool	Manage certificates for Axoflow components (Backend API, HTTP Proxy)
FilterX Validator	`axoflow`	Backend API	Validates FilterX expressions used in data processing pipelines
Traefik Ingress Controller	`ingress-traefik`	HTTP Proxy	Route the HTTP traffic between multiple Frontend/Backend APIs

Troubleshooting

- If you get the `[ERROR] Timed out waiting for k3s node to be Ready` error message, increase the timeout limit by running `K3S_TIMEOUT=180`, then rerun the installation script.

In case of other errors, run the following command to create a debug package from the cluster, and contact our support team.

```
sudo k3s kubectl cluster-info dump --all-namespaces  --output-directory=./cluster-dump && tar -czvf cluster-dump.tar.gz cluster-dump
```

Uninstall AxoConsole

To uninstall AxoConsole, run the installer with the `--uninstall` flag:

```
sudo ./axoflow-install.sh --uninstall
```

The uninstaller prompts before each step and requires separate confirmation for:

1. Removing the AxoConsole workloads and manifests
2. Deleting stored data (namespace and PersistentVolumeClaims) – **irreversible**

   > **CAUTION:**
   >
   > Hazard of data loss!
   >
   > Confirming step 2 permanently deletes all stored data.

3. Removing k3s

> **CAUTION:**
>
> Hazard of data loss!
>
> If you're using k3s to run other software apart from AxoConsole, decline step 3 to keep k3s in place.

The uninstaller is safe to run on a partial or incomplete installation.

18 Axoflow reference

Reference documentation for Axoflow: command-line tools, the query language, the glossary, and the message, metrics, and storage schemas.

Use these pages to look up exact details while you configure or query Axoflow.

- Message schema: the fields of the messages, which you can use in processing steps and search.
- Metrics schema: the labels you can use on the analytics pages and in log tapping filters.
- Storage schema: the fields you can search in your stores.
- AQL operator reference: the operators of AQL expressions.
- Global search: how to jump to any page or object in AxoConsole.
- Manual pages: the command-line tools, such as `axorouter-ctl`.
- Glossary: the terms used in Axoflow.

18.1 Global search

Jump to any AxoConsole page, pipeline object, or integration, or start creating a new object, from a single search field.

Global search is the quickest way to move around in AxoConsole. Type a few characters, then select a result to open it.

Open global search

To open global search, do one of the following:

- Select the search field in the page header of AxoConsole.
- Press `Ctrl + K` (Windows and Linux) or `⌘ + K` (macOS).

How results are matched

- Global search uses fuzzy matching on the title of the item. For objects, the title is the name of the object. If an object has no name, the title is its ID.
- Results are grouped by type. Each result shows the description of the object under its title.
- AxoConsole refreshes the list of objects each time you open global search, so objects you created recently are included.

What you can find

Selecting a result opens the item. For most objects, this is the overview or details page of the object.

Pages

The **Pages** group contains the main pages of AxoConsole: **Dashboard**, **Integrations**, **Analytics**, **Search Logs**, **Flows**, **Custom Steps**, **Topology**, **Sources**, **Collection Rules**, **Forwarding Rules**, **Routers**, **Connector Rules**, **Stores**, **Destinations**, **Provisioning**, **Rehydration**, **Alerting**, **Activity Logs**, **Roles**, **Axoflow** (version details and third-party licenses), and **Usage Metrics**.

Individual rehydration jobs don't appear in the results. To find a job, open the **Rehydration** page. For details, see Rehydrate data.

Objects

The objects of your pipeline are grouped by type. The group titles are the following:

- **Sources**: Source appliances. Opens the overview page of the source. See Sources.
- **Clusters**: Kubernetes clusters. Opens the overview page of the cluster.
- **Edges**: Edge hosts. Opens the overview page of the host. See Hosts.
- **Routers**: AxoRouter hosts. Opens the overview page of the router. See AxoRouter.
- **Destinations**: Destinations that receive data from your pipeline. Opens the overview page of the destination. See Destinations.
- **Flows**: Flows that route and process data. Opens the overview page of the flow. See Manage flows.
- **Collection rules**: Data collection rules of Axoflow agent hosts. Opens the **Collection rules** list, filtered to the selected rule. See Edge collection rules.
- **Connector rules**: Connector rules of AxoRouter hosts. Opens the **Connector rules** list, filtered to the selected rule. See AxoRouter connector rules.
- **Forwarding rules**: Rules that forward data from edge hosts to routers. Opens the overview page of the rule. See Data forwarding from edge hosts.
- **Store**: AxoStore and AxoLake stores. Opens the overview page of the store. See Storage.
- **Provisioning**: Active deploy commands. Opens the **Provisioning** page for the selected deploy command. See Provision pipeline elements.
- **Custom steps**: Custom processing steps that group processing steps into a reusable step. Opens the overview page of the step. See Processing steps.
- If detection is enabled for your organization, the **Rules** (Sigma detection rules), **Policies** (detection policies), and **Finding Flows** groups are also available.

Create actions

Every object group also contains a **Create** action, for example, **Create source**, **Create router**, **Create flow**, or **Create destination**. Select the action to open the dialog or page that creates that type of object. For edges, routers, and clusters, the action opens the **Provisioning** page, where you can generate an install command.

To start creating any object, open global search and type `create`.

Integrations

The **Integrations** group contains the products of the integrations catalog. You can search for the name of the vendor or the product. A product has a separate result for each capability it supports: **Source**, **Connector rule**, **Collection rule**, and **Destination**. Select a result to open the details of the integration on the **Integrations** page. See Integrations.

Global search and other search features

AxoConsole has three different search features:

- **Global search** finds and opens pages, objects, and integrations. It doesn't search your log data.
- **The filter bar** filters the list on the current page, for example, the hosts on the **Routers** page. For the query syntax, see AQL operator reference.
- **Log search** queries the data stored in your stores. See Search.

18.2 AQL operator reference

The comparison operators you can use in AQL expressions when you search or filter data, alerts, and analytics in AxoConsole.

AQL Query search supports the following comparison operators.

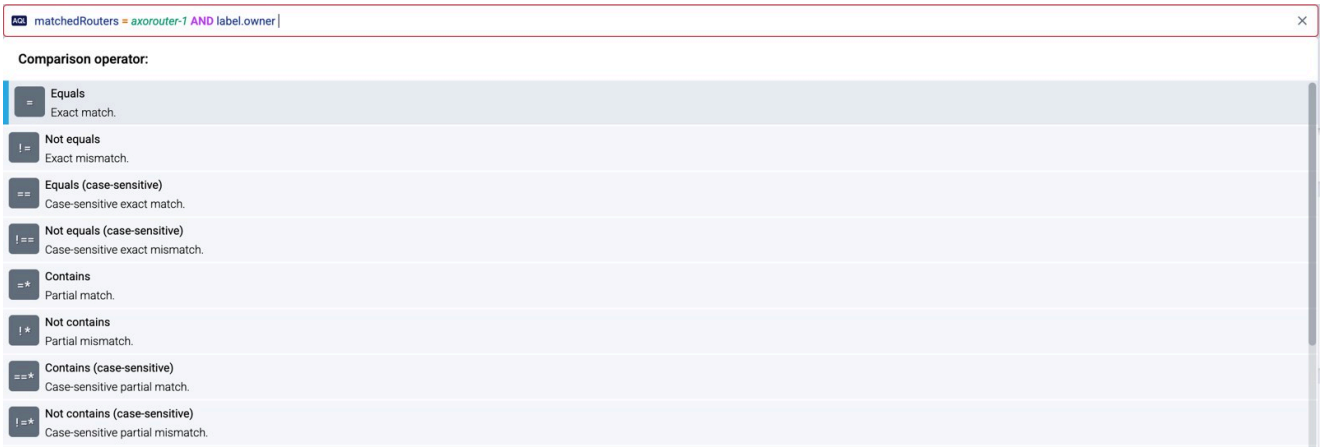
Case insensitive:

- Equals (`=`): the whole value equals to the pattern
- Not equals (`!=`): the value isn't exactly equal to the whole pattern
- Contains (`=*`): the value contains the given pattern
- Doesn't contain (`!*`): the value doesn't contain the given pattern
- Matches (`=~`): the pattern as a regular expression matches the value
- Doesn't match (`!~`): the case-insensitive regular expression doesn't match

The comparison operators have their corresponding case sensitive (strict) versions:

- Equals (`==`)
- Not equals (`!==`)
- Contains (`==*`)
- Doesn't contain (`!=*`)
- Matches: (`==~`)
- Doesn't match (`!=~`)

The syntax of the regular expressions accepted is the same general syntax used by Perl, Python, and other languages. The regular expressions are evaluated in case-insensitive mode in case of the `=~` and `!~` operators. The patterns are not anchored to the whole string, but you can use `^` at the beginning of the pattern and `$` at its end to match the whole value.



You can create complex queries using the `AND` and `OR` logic operators and parentheses, for example, `( host_name =* azure AND host_label_team = network ) OR ( host_name =* app AND host_label_app =* windows )`

Escaping rules

Enclose the field names and values in single-quotes (`'`), double-quotes (`"`), or `\`` if it contains characters not on this list: `@` , `a-z` , `0-9` , `._-` If all three quote types occur, enclose with single-quotes and escape single-quotes as `\\'` .

You can escape backslashes as `\\\\` .

18.3 Glossary

A standardized list of Axoflow terminology.

18.4 Manual pages

Read the manual pages of Axoflow command-line tools, such as axorouter-ctl.

18.4.1 axorouter-ctl

Commands and options of axorouter-ctl, the tool you run on an AxoRouter host to control and troubleshoot it.

Name

`axorouter-ctl` – Utility to control and troubleshoot AxoRouter

Synopsis

```
axorouter-ctl [command] [options]
```

Description

The `axorouter-ctl` application is a utility that is needed mainly for troubleshooting. For example, it can be used to:

- enable/disable troubleshooting messages
- display statistics about the processed messages
- display version information of AxoRouter.

Control syslog container

Usage: `axorouter-ctl control syslog <command> <options>`

This command allows you to access and control the syslog container of AxoRouter. Essentially, it's a wrapper for the `syslog-ng-ctl` command of AxoSyslog. Usually, you don't have to use this command unless our support team instructs you to.

For example:

- Enable debug messages for troubleshooting: `axorouter-ctl control syslog log-level verbose`
- Display the current configuration: `axorouter-ctl control syslog config --preprocessed`
- Reload the configuration: `axorouter-ctl control syslog reload`

For details, see the AxoSyslog documentation.

Show help

Usage: `axorouter-ctl help`

Lists the available commands. Running `axorouter-ctl` without a command shows the same list.

Show logs

Usage: `axorouter-ctl logs <target> <journalctl-options>`

Displays the logs of the target container: `cloudconnector`, `logstore`, `resolver`, `rsigma`, `store`, `syslog`, or `wec`. Cloud connector logs are available only if the AxoRouter host has a Axoflow Cloud Connector configured. WEC logs are available only if the AxoRouter host has a Windows Event Collector (WEC) connector configured.

The `axorouter-ctl logs` command is a wrapper for the `journalctl` command, so you can add regular journalctl options to the command. For example, `-n 50` to show the 50 most recent lines: `axorouter-ctl logs syslog -n 50`

Prune images

Usage: `axorouter-ctl prune`

Removes all unused images from the local store, similarly to `podman-image-prune`.

Pull images

Usage: `axorouter-ctl pull`

Download images needed for AxoRouter. This is usually not needed unless you have a custom environment, for example, proxy settings or pull credentials that aren't available when the AxoRouter container is run.

Manage the `axorouter-resolver` service

Usage: `axorouter-ctl resolver <command>`

Manages the `axorouter-resolver` service. Use the `clear-cache` command to delete the cache of the service.

Shell access to AxoRouter

Usage: `axorouter-ctl shell <target>`

Open a shell into the target container (`logstore` , `resolver` , `rsigma` , `store` , `syslog` , or `wec`). Usually needed only for troubleshooting purposes, when requested by our support team.

Statistics and metrics

Usage: `axorouter-ctl stats syslog [legacy|prometheus]`

Display the statistics and metrics of the syslog container in Prometheus or legacy format, from the underlying AxoSyslog process. If you don't specify the format, the command uses the legacy format. For details, see the AxoSyslog documentation.

Access AxoStore database

Usage: `axorouter-ctl store`

> **CAUTION:**
>
> Access the database directly only when instructed by the support team, or if you know exactly what you're doing.

Start an interactive SQL session using the command-line ClickHouse client to access data stored in AxoStore. Type `exit` to end the session.

Version

Usage: `axorouter-ctl version`

Show the version number of AxoRouter and of the underlying AxoSyslog build, for example:

```
0.58.0
axosyslog 4 (4.11.0)
Config version: 4.2
Installer-Version: 4.11.0
```

Wipe disk buffer files

Usage: `axorouter-ctl wipe [-n|--dry-run] [-a|--all] [--backup-dir <directory>]`

Cleans up the corrupted and abandoned disk buffer files. Usually, you don't have to use this command unless our support team instructs you to, or an alert suggests it.

How the command handles a file depends on the state of the file:

- Corrupted files are moved to the backup directory. The command never deletes them.
- Abandoned files that contain no messages are deleted.
- Abandoned files that contain messages are truncated, then moved to the backup directory. To delete these files without keeping a backup, use the `-a` or `--all` option.

To see what the command would move, truncate, and delete without changing anything, use the `-n` or `--dry-run` option.

A backup keeps the original path of the file, with a `.backup` suffix. For example, `/var/lib/syslog-ng/syslog-ng-00000.rqf` is backed up as `<backup-directory>/var/lib/syslog-ng/syslog-ng-00000.rqf.backup`.

The default backup directory is `/var/lib/syslog-ng/diskqueue_backup` in the syslog container. On the host, this is the `diskqueue_backup` directory in the persist directory of AxoRouter. To use a different backup directory, use the `--backup-dir <directory>` option. The directory is a path in the syslog container: only directories under `/var/lib/syslog-ng` are kept when the container restarts.

The command prints a warning, and still runs, if:

- the backup directory isn't under `/var/lib/syslog-ng`, because the backups are lost when the container is recreated, or
- the persistent storage directory on the host isn't `/var/lib/syslog-ng`. In this case, the warning shows the directory on the host where the backups are.

If the command processed abandoned disk buffer files, it reloads AxoSyslog when the cleanup is complete.

Environment variables

`AVOID_SUDO`

Set to `1` to prevent `axorouter-ctl` from running itself with `sudo`. By default, the command runs itself again as root if you start it as an unprivileged user.

`AXOROUTER_X`

Set to any non-empty value to print every command that `axorouter-ctl` runs. Use it to troubleshoot `axorouter-ctl` itself.

18.5 Message schema

Learn how Axoflow structures messages, and look up the fields of the Axoflow message schema.

18.5.1 Overview

How AxoRouter maps incoming data to its OpenTelemetry-based message model, and where you can use the meta fields.

When processing incoming data, AxoRouter automatically converts everything to its internal message model, and maps the contents of the model to the specific destinations as needed. The internal message model of AxoRouter is based on the OpenTelemetry Log Data Model, so what is sent to the destination is determined by the `log` section of the model (the main payload is `log.body`).

The `meta` section contains all the information and metadata AxoRouter has about the message: what was its source, how it was received, where it will be forwarded, how was it classified, and so on. You can use these metadata for example:

- in the processing steps of the flows,
- on the search interfaces of AxoConsole,
- during Flow tapping and Log tapping.

The related metrics of these data can be used:

- on the analytics pages, and
- in Flow tapping and log tapping filters.

When AxoRouter sends the message to its destination, parts of these metadata is automatically mapped into the relevant `log` field, but most of it (for example, labels) isn't send to the destination by default.

For most destinations, there are specific fields that allow you to configure specific values that are sent to the destination, or to override the `log.body` field completely.

The message model has the following main elements:

- `log`: The `log` data structure describes the log record.
- `meta`: Metadata about a specific message record, for example, a log message.
- `resource`: The `resource` data structure describes the resource that generated the log record.
- `scope`: The `scope` data structure describes the Log Scope.

18.5.2 Message schema reference

Look up the fields of the Axoflow message model, including the meta fields you can use in processing steps and search.

Expand all

Log (object)

log

The `log` data structure describes a log record.

Log Attributes (object)

log > attributes

Additional attributes that describe the specific event occurrence. Every attribute key must be unique.

Log Body (string)

log > body

The body of the log record, which can contain strings and structured data composed of arrays and maps of other values.

Observed Time (datetime)

log > observed_time_unix_nano

Time when the event was observed by the data pipeline, in UNIX Epoch time (nanoseconds elapsed since 00:00:00 UTC on 1 January 1970). A value of `0` indicates unknown or missing timestamp.

- For events that originate in OpenTelemetry, this timestamp is typically set at the generation time and is equal to `time_unix_nano`.
- For events originating externally and collected by an Axoflow agent or an AxoRouter, this is the time when the Axoflow pipeline observed the event.

Log Severity Number (number)

log > severity_number

Numerical value of the severity, normalized to values described in Log Data Model.

Log Severity Text (string)

log > severity_text

The severity as a string (log level). The original string representation as described at the source. For the numerical to string mapping, see log.severity_number.

Event date (datetime)

log > time_unix_nano

The time when the event occurred in UNIX Epoch time (nanoseconds elapsed since 00:00:00 UTC on 1 January 1970). A value of `0` indicates unknown or missing timestamp.

Metadata (object)

meta

Metadata about a specific message record, for example, a log message.

Classified (object)

meta > classified

The value of vendor, product, service name and other metadata set during classification.

CrowdStrike metadata identified during classification (object)

meta > classified > crowdstrike

CrowdStrike Falcon Next-Gen SIEM metadata extracted by the classification adapters.

CrowdStrike Event Module (string)

meta > classified > crowdstrike > module

The value of the CrowdStrike Parsing Standard `event.module` field. CrowdStrike expects the bare product name here, for example, `ios` for Cisco IOS. When unset, CrowdStrike normalization keeps the ECS-normalized `event.module` value.

CrowdStrike Vendor (string)

meta > classified > crowdstrike > vendor

The value of the CrowdStrike Parsing Standard `#Vendor` tag. CrowdStrike expects its own short, lowercase vendor name here, for example, `paloalto`. For details, see the CrowdStrike vendor guidelines. CrowdStrike normalization resolves the tag from `meta.destination.crowdstrike.vendor`, then this field, then `meta.vendor`.

Google SecOps metadata identified during classification (object)

meta > classified > google_secops

Google SecOps metadata extracted by the classification adapters.

Google SecOps Log Type (string)

meta > classified > google_secops > log_type

The classified Google Secops log type.

Classified Product (string)

meta > classified > product

The product name of the source that generated the message, as determined by the classification.

Related metric label: `auto_classified_product`

Classified Service Metadata (object)

meta > classified > service

The name of the service that generated the message, as determined by the classification.

Classified Service Name (string)

meta > classified > service > name

Classified service Name for appliance.

Related metric label: `auto_classified_service`

Sigma metadata identified during classification (object)

meta > classified > sigma

Sigma metadata extracted by the classification adapters.

Sigma Category (string)

meta > classified > sigma > category

The classified Sigma logsource category, for example, `webserver`, `proxy`, `firewall`, `antivirus`, or `dns`.

Sigma Product (string)

meta > classified > sigma > product

The classified Sigma logsource product.

Sigma Service (string)

meta > classified > sigma > service

The classified Sigma service.

Splunk metadata (object)

meta > classified > splunk

Splunk metadata extracted during classification.

Host (string)

meta > classified > splunk > host

The name of the host as sent to Splunk. Usually, this is the hostname of the source where the data originated from.

Splunk Index (string)

meta > classified > splunk > index

The name of the Splunk index where the message is sent. The index must exist in Splunk, otherwise sending the data will fail.

Splunk Source (string)

meta > classified > splunk > source

The source field sent to Splunk, containing where the event originated. For example, the protocol and port for network-based sources, or the path and filename for log files.

Splunk Sourcetype (string)

meta > classified > splunk > sourcetype

The Splunk sourcetype value that corresponds to the appliance, application, or service that generated the data.

Classified Vendor (string)

meta > classified > vendor

The name of the vendor of the source that generated the message, as determined by the classification.

Related metric label: `auto_classified_vendor`

XSIAM metadata identified during classification (object)

meta > classified > xsiam

XSIAM metadata extracted by the classification adapters.

XSIAM Product (string)

meta > classified > xsiam > product

The classified XSIAM product.

XSIAM Vendor (string)

meta > classified > xsiam > vendor

The classified XSIAM vendor.

Connection (object)

meta > connection

Information about the network connection that transmitted the message.

Destination IP address (string)

meta > connection > dest_ip

IP address where the message was sent to according to the IP header.

Related metric label: `dest_ip`

Destination port (number)

meta > connection > dest_port

TCP or UDP port number where the message was sent to according to the transport header.

Related metric label: `dest_port`

IP protocol (number)

meta > connection > ip_protocol

Network protocol version used for receiving this message.

Possible values: `4, 6`

Related metric label: `ip_protocol`

Protocol (string)

meta > connection > protocol

Transport protocol used for receiving this message.

Possible values: `TCP, UDP`

Related metric label: `protocol`

Source IP address (string)

meta > connection > src_ip

IP address that sent the message according to the IP header.

Related metric label: `src_ip`

Transport (string)

meta > connection > transport

The transport mechanism used to retrieve or receive the message.

Example: `rfc3164+tls`

Related metric label: `transport`

Connector (object)

meta > connector

Information about the Connector that received the log. See the Connectors section on the **Overview** page of the router for details.

Labels (object)

meta > connector > labels

Labels set on the connector that received the message.

Prefix of related metric labels: `connector_label_*`

Name (string)

meta > connector > name

The name of the connector that received the log. The name of the connector instance follows the `<name-of-AxoRouter>-<suffix>` schema, or `<name-of-AxoRouter>-<name-of-connector-rule>` if the rule has no suffix. For details, see Connector rules.

Example: `myrouter-otlp`

Related metric label: `connector_name`

Port (string)

meta > connector > port

The port number of the connector that received the message. Empty if the connector has no listening port.

Example: `514`

Syslog Connector Related Metadata (object)

meta > connector > syslog

Parameters for processing the message from a Syslog source.

Splunk Log Type (string)

meta > connector > syslog > splunk_log_type

The type of the payload received from Splunk HF.

Possible values: `syslog, plaintext`

Type (string)

meta > connector > type

The type of the connector that received the message.

Possible values: `awsCloudTrail, awsVPCFlowLogs, azureEventHubs, crowdstrikeNGSIEM, idira, otlp, soup, syslog, tenableAuditLog, webhook, windowsEvents`

Example: `otlp`

Related metric label: `connector_type`

Destination Connector (object)

meta > destination

Information about the Destination Connector.

AxoStore Metadata (object)

meta > destination > axostore

Parameters for sending the message to an AxoStore destination.

AxoStore Raw Body (boolean)

meta > destination > axostore > body_raw

Override the option to send the raw message body to the AxoStore destination. (default: false)

AxoStore Raw Meta (boolean)

meta > destination > axostore > meta_raw

Override the option to send raw metadata to AxoStore destination. (default: true)

Azure Monitor Metadata (object)

meta > destination > azuremonitor

Parameters for sending the message to an Azure Monitor destination.

Azure Monitor Log Message (object)

meta > destination > azuremonitor > message

Override the message sent to Azure Monitor. When set, `log.body` is ignored.

Azure Monitor Stream Name (string)

meta > destination > azuremonitor > stream_name

Override the destination stream name in Azure Monitor. When set, `stream-name` configuration option is ignored.

ClickHouse Metadata (object)

meta > destination > clickhouse

Parameters for sending the message to a ClickHouse destination.

Prefix of related metric labels: `clickhouse_*`

ClickHouse Raw Body (boolean)

meta > destination > clickhouse > body_raw

Override the option to send raw body to the ClickHouse destination. (default: false)

ClickHouse Raw Meta (boolean)

meta > destination > clickhouse > meta_raw

Override the option to send raw metadata to the ClickHouse destination. (default: true)

CrowdStrike NG-SIEM Metadata (object)

meta > destination > crowdstrike

Parameters for sending the message to a CrowdStrike Next-Gen SIEM destination.

Prefix of related metric labels: `crowdstrike_*`

CrowdStrike Event (string)

meta > **destination** > **crowdstrike** > **event**

Override the raw event (`@rawstring`) sent to CrowdStrike NG-SIEM. When set, the original log body is not sent.

CrowdStrike Fields (object)

meta > **destination** > **crowdstrike** > **fields**

Fields to send to CrowdStrike NG-SIEM. This overrides the default fields set by the destination connector.

Send Normalized (boolean)

meta > **destination** > **crowdstrike** > **send_normalized**

Whether to send the CrowdStrike-normalized form of the log in the HEC `fields` member. Overrides the setting of the destination connector, which defaults to `true` .

Send Raw (boolean)

meta > **destination** > **crowdstrike** > **send_raw**

Whether to send the log in its original text form as the HEC `event` member. Overrides the setting of the destination connector, which defaults to `true` .

CrowdStrike Time (integer)

meta > **destination** > **crowdstrike** > **time**

Override the event time sent to CrowdStrike NG-SIEM, as epoch milliseconds. When set, `log.time_unix_nano` is ignored. The value must be an integer, because CrowdStrike silently drops non-integer time values.

CrowdStrike Vendor Tag (string)

meta > **destination** > **crowdstrike** > **vendor**

CrowdStrike normalization resolves the tag from this field, then `meta.classified.crowdstrike.vendor`, then `meta.vendor`.

Dynatrace Metadata (object)

meta > destination > dynatrace

Parameters for sending the message to a Dynatrace destination.

Dynatrace Extra Fields (object)

meta > destination > dynatrace > extra_fields

Extra fields to add to the formatted Dynatrace Log Message.

Dynatrace Log Message (object)

meta > destination > dynatrace > log_message

Override the message sent to Dynatrace. When set, `log.body` is ignored.

Elasticsearch Metadata (object)

meta > destination > elasticsearch

Parameters for sending the message to an Elasticsearch destination.

Prefix of related metric labels: `elasticsearch_*`

Elasticsearch Fields (object)

meta > destination > elasticsearch > fields

Fields to send to Elasticsearch. This overrides the default fields set by the destination connector.

Elasticsearch Index (string)

meta > destination > elasticsearch > index

The Elasticsearch index to send the message to.

Elasticsearch Message (string)

meta > destination > elasticsearch > message

Override the message to send to Elasticsearch. When set, `log.body` is ignored.

Elasticsearch Timestamp (string)

meta > destination > elasticsearch > timestamp

Override the timestamp to send to Elasticsearch. When set, `log.time_unix_nano` is ignored.

Google SecOps Metadata (object)

meta > destination > google_secops

Parameters for sending the message to a Google SecOps destination.

Prefix of related metric labels: `googlesecops_*`

Google SecOps Customer ID (string)

meta > destination > google_secops > customer_id

This overrides the default customer ID configured in the destination connector.

Google SecOps Labels (object)

meta > destination > google_secops > labels

Labels to send to Google SecOps. This overrides the default labels set by the destination connector.

Google SecOps Log Text (string)

meta > destination > google_secops > log_text

The text of the message to send to Google SecOps. When set, `log.body` is ignored.

Google SecOps Log Type (string)

meta > destination > google_secops > log_type

This overrides the default log type configured in the destination connector.

Google SecOps Namespace (string)

meta > destination > google_secops > namespace

This overrides the default namespace configured in the destination connector.

Google SecOps Timestamp (string)

meta > destination > google_secops > ts_rfc3339

The timestamp of the message in RFC3339 format. When set, `log.time_unix_nano` is ignored.

Labels (object)

meta > destination > labels

Labels set on the connector.

Logstore Metadata (object)

meta > destination > logstore

Parameters for sending the message to a Logstore destination.

Logstore Raw Body (boolean)

meta > destination > logstore > body_raw

Override the option to send the raw message body to the Logstore destination. (default: false)

Logstore Raw Meta (boolean)

meta > destination > logstore > meta_raw

Override the option to send raw metadata to the Logstore destination. (default: true)

Message Format (string)

meta > destination > message_format

Possible values: `text, parsed, normalized_ecs, normalized_ocsf, normalized_xdm, xsiam_dataset`

Name (string)

meta > destination > name

Name of the destination where AxoRouter sent the message.

Example: `myrouter-splunk`

Related metric label: `destination_name`

OpenObserve Metadata (object)

meta > destination > openobserve

Parameters for sending the message to an OpenObserve destination.

Prefix of related metric labels: `openobserve_*`

OpenObserve Fields (object)

meta > destination > openobserve > fields

Fields to send to OpenObserve. This overrides the default fields set by the destination connector.

OpenObserve Message (string)

meta > destination > openobserve > message

Override the message to send to OpenObserve. When set, `log.body` is ignored.

OpenObserve Organization (string)

meta > destination > openobserve > organization

The OpenObserve organization to send the message to.

OpenObserve Stream (string)

meta > destination > openobserve > stream

The OpenObserve stream to send the message to.

OpenObserve Timestamp (string)

meta > destination > openobserve > timestamp

Override the timestamp to send to OpenObserve. When set, `log.time_unix_nano` is ignored.

Google Pub/Sub Metadata (object)

meta > destination > pubsub

Parameters for sending the message to a Google PubSub destination.

Prefix of related metric labels: `pubsub_*`

Google Pub/Sub Attributes (object)

meta > destination > pubsub > attributes

Override the attributes key-value pairs for the Pub/Sub Event.

Google Pub/Sub Data (string)

meta > destination > pubsub > data

Override the data to send to Google Pub/Sub. When set, `log.body` is ignored.

Google Pub/Sub Project (string)

meta > destination > pubsub > project

The ID of the Google Cloud project where the data is sent.

Google Pub/Sub Topic (string)

meta > destination > pubsub > topic

The name of the Google Pub/Sub topic to send the data to.

RSigma Metadata (object)

meta > destination > rsigma

Parameters for sending the message to an RSigma destination.

RSigma Log Message (object)

meta > destination > rsigma > message

Override the message sent to RSigma. When set, `log.body` is ignored.

Sigma Product (string)

meta > destination > rsigma > product

Override the product value sent to RSigma.

Sigma service (string)

meta > destination > rsigma > service

Override the service value sent to RSigma.

S3 Metadata (object)

meta > destination > s3

Parameters for sending the message to a S3 destination.

S3 Record (string)

meta > destination > s3 > record

Override the message sent to S3. When set, `log.body` is ignored.

Security Lake Metadata (object)

meta > destination > security_lake

Parameters for sending the message to a Security Lake destination.

Security Lake Message (string)

meta > destination > security_lake > message

Override the message sent to Security Lake. When set, `log.body` is ignored.

Splunk metadata (object)

meta > destination > splunk

Parameters for sending the message to a Splunk destination.

Prefix of related metric labels: `splunk_*`

Event (string)

meta > destination > splunk > event

The raw event sent to Splunk. Overrides `log.body` and avoids automatic formatting completely.

Splunk Fields (object)

meta > destination > splunk > fields

Fields to send to Splunk. This overrides the default fields set by the destination connector.

Host (string)

meta > destination > splunk > host

The name of the host as sent to Splunk. Usually, this is the hostname of the source where the data originated from.

Splunk Index (string)

meta > destination > splunk > index

The name of the Splunk index where the message is sent. The index must exist in Splunk, otherwise sending the data will fail.

Splunk Source (string)

meta > destination > splunk > source

The source field sent to Splunk, containing where the event originated. For example, the protocol and port for network-based sources, or the path and filename for log files.

Splunk Sourcetype (string)

meta > destination > splunk > sourcetype

The Splunk sourcetype value that corresponds to appliance, application, or service that generated the data.

Splunk Time (string)

meta > destination > splunk > time

Override the time sent to Splunk. When set, log.time_unix_nano is ignored.

Sumo Logic Metadata (object)

meta > destination > sumologic

Parameters for sending the message to a Sumo Logic destination.

Sumo Logic Fields (object)

meta > destination > sumologic > fields

Fields to send to Sumo Logic. This overrides the default fields set by the destination connector.

Sumo Logic Message (string)

meta > destination > sumologic > message

Override the message sent to Sumo Logic. When set, `log.body` is ignored.

Sumo Logic Source Category (string)

meta > destination > sumologic > source_category

Override the source category configured in the destination connector.

Sumo Logic Source Name (string)

meta > destination > sumologic > source_name

Override the source name configured in the destination connector.

Syslog metadata (object)

meta > destination > syslog

Parameters for sending the message to a Syslog destination.

Syslog Message (string)

meta > destination > syslog > message

Override the entire payload sent to Syslog, including the envelope. When set, `log.body` is ignored.

Type (string)

meta > destination > type

Type of the destination, for example, `splunkHEC`.

Possible values: `azuremonitor, elasticsearch, openObserve, router, opentelemetry, pubsub, s3, splunkHEC, googleSecOps, sumologic, dynatrace, xsiam, syslog, axostore, clickhouse, sink, crowdstrikeNGSIEM`

Example: `splunkHEC`

Related metric label: `destination_type`

XSIAM Metadata (object)

meta > destination > xsiam

Parameters for sending the message to an XSIAM destination

XSIAM Log Message (object)

meta > destination > xsiam > message

Override the message sent to XSIAM. When set, `log.body` is ignored.

XSIAM Product (string)

meta > destination > xsiam > product

Override the product value sent to XSIAM.

XSIAM Vendor (string)

meta > destination > xsiam > vendor

Override the vendor value sent to XSIAM.

Edge (object)

meta > edge

Information about the edge host and its resources that sent the message.

Connector metadata (object)

meta > edge > connector

Metadata about the edge connector that sent the message.

Labels (object)

meta > edge > connector > labels

Labels set on the edge connector that sent the message.

Prefix of related metric labels: `edge_connector_label_*`

Name (string)

meta > edge > connector > name

The name of the edge connector that sent the message.

Related metric label: `edge_connector_name`

Connector Rule ID (string)

meta > edge > connector > rule_id

The ID of the owner ConnectorRule resource in Axoflow that created the edge connector.

Related metric label: `edge_connector_rule_id`

Type (string)

meta > edge > connector > type

The type of the edge connector that sent the message.

Possible values: `windowsEventLog, windowsEventTracing, otelFile, otelJournald`

Example: `otelFile`

Related metric label: `edge_connector_type`

Flow metadata (object)

meta > edge > flow

Metadata about the edge forwarding rule that sent the message.

Name (string)

meta > edge > flow > name

The name of the edge forwarding rule that sent the message.

Related metric label: `edge_flow_name`

Envelope (object)

meta > envelope

Application metadata parsed from the envelope.

Elasticsearch Bulk metadata extracted from the message envelope (object)

meta > envelope > elasticsearch

Metadata from the Elasticsearch Bulk API action line that accompanied the document.

Bulk Operation (string)

meta > envelope > elasticsearch > operation

The Bulk API operation from the action line, for example, `index` or `create`.

Bulk Action Parameters (object)

meta > envelope > elasticsearch > params

The parameters of the action line, for example, `_index` and `_id`.

Envelope Extracted Service metadata (object)

meta > envelope > service

Envelope Extracted service info about appliances.

Envelope Extracted Service Name (string)

meta > envelope > service > name

Envelope Extracted Service Name for appliance.

Splunk metadata extracted from the message envelope (object)

meta > envelope > splunk

Splunk metadata received from a Heavy Forwarder in the Axoflow envelope.

Host (string)

meta > envelope > splunk > host

The name of the host as received from Splunk. Usually, this is the hostname of the source where the data originated from.

Splunk Index (string)

meta > envelope > splunk > index

The name of the Splunk index where the message is received from.

Splunk Source (string)

meta > envelope > splunk > source

The source field received from Splunk, containing where the event originated. For example, the protocol and port for network-based sources, or the path and filename for log files.

Splunk Sourcetype (string)

meta > envelope > splunk > sourcetype

The Splunk sourcetype value that corresponds to appliance, application, or service that generated the data.

Message envelope type (string)

meta > envelope > type

Type of the message envelope. AxoRouter sets `json`, `leef`, `xml`, or `cef` when it detects such a payload.

Fallback Failure Info (array)

meta > fallback_failure_info

Information on why the message was routed to the fallback flow due to errors or filtering across all other defined flows.

Flow name (string)

meta > flow

Name of the flow processing the message.

Related metric label: `flow`

Host (object)

meta > host

Information about the host that sent the message.

Source Host Labels (object)

meta > host > labels

The labels set in the inventory for the host the message originates from. Note that if the host is sending data to an AxoRouter connector that doesn't perform automatic classification, then changing the product and vendor labels can affect the final metadata in the destination, for example, the sourcetype assigned to the data in Splunk.

Prefix of related metric labels: `host_label_*`

Source Host Name (string)

meta > host > name

The name of the host the message originates from (based on the inventory).

Related metric label: `host_name`

Host Candidate (object)

meta > host_candidate

Device ID (string)

meta > host_candidate > id

Device ID found in the message.

Related metric label: `host_candidate_id`

IP Address (string)

meta > host_candidate > ip

IP address found in the message.

Related metric label: `host_candidate_ip`

Host Name of the Last Hop (string)

meta > host_candidate > last_hop_name

Host field found in the message's envelope (which can be either the subject of the message, or the name of the host that forwarded it).

Related metric label: `host_candidate_last_hop_name`

Host Name (string)

meta > host_candidate > name

Host name found in the message.

Related metric label: `host_candidate_name`

Kubernetes (object)

meta > kubernetes

Kubernetes metadata received from the collector.

Container Name (string)

meta > kubernetes > container

Kubernetes container name found in the message metadata.

Related metric label: `kubernetes_container`

Namespace (string)

meta > kubernetes > namespace

Name of the Kubernetes namespace found in the message metadata.

Related metric label: `kubernetes_namespace`

Product (string)

meta > product

The product name of the appliance, application, or service that generated the message.

Related metric label: `product`

Router (object)

meta > router

The name and labels of the AxoRouter instance that processed the message.

Labels (object)

meta > router > labels

Labels of the AxoRouter instance that processed the message.

Prefix of related metric labels: `axo_host_label_*`

Router Name (string)

meta > router > name

The hostname of the AxoRouter instance that processed the message.

Related metric label: `axo_host_name`

Service metadata (object)

meta > service

Information about the service that generated the message.

Service Name (string)

meta > service > name

Name of the service that generated the message. For syslog messages, that's usually the value of the PROGRAM field.

Related metric label: `service`

Vendor (string)

meta > vendor

The vendor of the appliance, application, or service that generated the message.

Related metric label: `vendor`

Resource (object)

resource

The `resource` data structure describes the resource that generated the log record.

Resource Attributes (object)

resource > attributes

Attributes that describe the resource. Every attribute key must be unique.

Scope (object)

scope

Log Scope representation

Scope Attributes (object)

scope > **attributes**

Attributes that describe the log scope. Every attribute key must be unique.

Name (string)

scope > **name**

Name of the log scope

Version (string)

scope > **version**

Version of the log scope

18.6 Metrics schema

The metrics labels you can use on the analytics pages and in log tapping filters, and the message fields they relate to.

The following list shows the metrics labels related to the message schema. You can use these metrics labels:

- on the analytics pages, and
- in log tapping filters.

auto_classified_product

Type: string
Related schema element: meta.classified.product

The product name of the source that generated the message, as determined by the classification.

auto_classified_service

Type: string
Related schema element: meta.classified.service.name

Classified service Name for appliance.

auto_classified_vendor

Type: string
Related schema element: meta.classified.vendor

The name of the vendor of the source that generated the message, as determined by the classification.

dest_ip

Type: string
Related schema element: meta.connection.dest_ip

IP address where the message was sent to according to the IP header.

dest_port

Type: number
Related schema element: meta.connection.dest_port

TCP or UDP port number where the message was sent to according to the transport header.

ip_protocol

Type: number
Related schema element: meta.connection.ip_protocol

Network protocol version used for receiving this message.

protocol

Type: string
Related schema element: meta.connection.protocol

Transport protocol used for receiving this message.

src_ip

Type: string
Related schema element: meta.connection.src_ip

IP address that sent the message according to the IP header.

transport

Type: string
Related schema element: meta.connection.transport

The transport mechanism used to retrieve or receive the message.

connector_label_labels

Type: object
Related schema element: meta.connector.labels

Labels set on the connector that received the message.

connector_name

Type: string
Related schema element: meta.connector.name

The name of the connector that received the log. The name of the connector instance follows the `<name-of-AxoRouter>-<suffix>` schema, or `<name-of-AxoRouter>-<name-of-connector-rule>` if the rule has no suffix. For details, see Connector rules.

connector_type

Type: string
Related schema element: meta.connector.type

The type of the connector that received the message.

clickhouse_clickhouse

Type: object
Related schema element: meta.destination.clickhouse

Parameters for sending the message to a ClickHouse destination.

clickhouse_body_raw

Type: boolean
Related schema element: meta.destination.clickhouse.body_raw

Override the option to send raw body to the ClickHouse destination. (default: false)

clickhouse_meta_raw

Type: boolean
Related schema element: meta.destination.clickhouse.meta_raw

Override the option to send raw metadata to the ClickHouse destination. (default: true)

crowdstrike_crowdstrike

Type: object
Related schema element: meta.destination.crowdstrike

Parameters for sending the message to a CrowdStrike Next-Gen SIEM destination.

crowdstrike_event

Type: string
Related schema element: meta.destination.crowdstrike.event

Override the raw event (`@rawstring`) sent to CrowdStrike NG-SIEM. When set, the original log body is not sent.

crowdstrike_fields

Type: object
Related schema element: meta.destination.crowdstrike.fields

Fields to send to CrowdStrike NG-SIEM. This overrides the default fields set by the destination connector.

crowdstrike_send_normalized

Type: boolean
Related schema element: meta.destination.crowdstrike.send_normalized

Whether to send the CrowdStrike-normalized form of the log in the HEC `fields` member. Overrides the setting of the destination connector, which defaults to `true` .

crowdstrike_send_raw

Type: boolean
Related schema element: meta.destination.crowdstrike.send_raw

Whether to send the log in its original text form as the HEC `event` member. Overrides the setting of the destination connector, which defaults to `true` .

crowdstrike_time

Type: integer
Related schema element: meta.destination.crowdstrike.time

Override the event time sent to CrowdStrike NG-SIEM, as epoch milliseconds. When set, `log.time_unix_nano` is ignored. The value must be an integer, because CrowdStrike silently drops non-integer time values.

crowdstrike_vendor

Type: string
Related schema element: meta.destination.crowdstrike.vendor

Override the value of the `#Vendor` tag sent to CrowdStrike NG-SIEM. CrowdStrike expects its own short, lowercase vendor name here, for example, `paloalto`. For details, see the CrowdStrike vendor guidelines. CrowdStrike normalization resolves the tag from this field, then `meta.classified.crowdstrike.vendor`, then `meta.vendor`.

elasticsearch_elasticsearch

Type: object
Related schema element: meta.destination.elasticsearch

Parameters for sending the message to an Elasticsearch destination.

elasticsearch_fields

Type: object
Related schema element: meta.destination.elasticsearch.fields

Fields to send to Elasticsearch. This overrides the default fields set by the destination connector.

elasticsearch_index

Type: string
Related schema element: meta.destination.elasticsearch.index

The Elasticsearch index to send the message to.

elasticsearch_message

Type: string
Related schema element: meta.destination.elasticsearch.message

Override the message to send to Elasticsearch. When set, `log.body` is ignored.

elasticsearch_timestamp

Type: string
Related schema element: meta.destination.elasticsearch.timestamp

Override the timestamp to send to Elasticsearch. When set, `log.time_unix_nano` is ignored.

googlesecops_google_secops

Type: object
Related schema element: meta.destination.google_secops

Parameters for sending the message to a Google SecOps destination.

googlesecops_customer_id

Type: string
Related schema element: meta.destination.google_secops.customer_id

This overrides the default customer ID configured in the destination connector.

googlesecops_labels

Type: object
Related schema element: meta.destination.google_secops.labels

Labels to send to Google SecOps. This overrides the default labels set by the destination connector.

googlesecops_log_text

Type: string
Related schema element: meta.destination.google_secops.log_text

The text of the message to send to Google SecOps. When set, `log.body` is ignored.

googlesecops_log_type

Type: string
Related schema element: meta.destination.google_secops.log_type

This overrides the default log type configured in the destination connector.

googlesecops_namespace

Type: string
Related schema element: meta.destination.google_secops.namespace

This overrides the default namespace configured in the destination connector.

googlesecops_ts_rfc3339

Type: string
Related schema element: meta.destination.google_secops.ts_rfc3339

The timestamp of the message in RFC3339 format. When set, `log.time_unix_nano` is ignored.

destination_name

Type: string
Related schema element: meta.destination.name

Name of the destination where AxoRouter sent the message.

openobserve_openobserve

Type: object
Related schema element: meta.destination.openobserve

Parameters for sending the message to an OpenObserve destination.

openobserve_fields

Type: object
Related schema element: meta.destination.openobserve.fields

Fields to send to OpenObserve. This overrides the default fields set by the destination connector.

openobserve_message

Type: string
Related schema element: meta.destination.openobserve.message

Override the message to send to OpenObserve. When set, `log.body` is ignored.

openobserve_organization

Type: string
Related schema element: meta.destination.openobserve.organization

The OpenObserve organization to send the message to.

openobserve_stream

Type: string
Related schema element: meta.destination.openobserve.stream

The OpenObserve stream to send the message to.

openobserve_timestamp

Type: string
Related schema element: meta.destination.openobserve.timestamp

Override the timestamp to send to OpenObserve. When set, `log.time_unix_nano` is ignored.

pubsub_pubsub

Type: object
Related schema element: meta.destination.pubsub

Parameters for sending the message to a Google PubSub destination.

pubsub_attributes

Type: object
Related schema element: meta.destination.pubsub.attributes

Override the attributes key-value pairs for the Pub/Sub Event.

pubsub_data

Type: string
Related schema element: meta.destination.pubsub.data

Override the data to send to Google Pub/Sub. When set, `log.body` is ignored.

pubsub_project

Type: string
Related schema element: meta.destination.pubsub.project

The ID of the Google Cloud project where the data is sent.

pubsub_topic

Type: string
Related schema element: meta.destination.pubsub.topic

The name of the Google Pub/Sub topic to send the data to.

splunk_splunk

Type: object
Related schema element: meta.destination.splunk

Parameters for sending the message to a Splunk destination.

splunk_event

Type: string
Related schema element: meta.destination.splunk.event

The raw event sent to Splunk. Overrides `log.body` and avoids automatic formatting completely.

splunk_fields

Type: object
Related schema element: meta.destination.splunk.fields

Fields to send to Splunk. This overrides the default fields set by the destination connector.

splunk_host

Type: string
Related schema element: meta.destination.splunk.host

The name of the host as sent to Splunk. Usually, this is the hostname of the source where the data originated from.

splunk_index

Type: string
Related schema element: meta.destination.splunk.index

The name of the Splunk index where the message is sent. The index must exist in Splunk, otherwise sending the data will fail.

splunk_source

Type: string
Related schema element: meta.destination.splunk.source

The source field sent to Splunk, containing where the event originated. For example, the protocol and port for network-based sources, or the path and filename for log files.

splunk_sourcetype

Type: string
Related schema element: meta.destination.splunk.sourcetype

The Splunk sourcetype value that corresponds to appliance, application, or service that generated the data.

splunk_time

Type: string
Related schema element: meta.destination.splunk.time

Override the time sent to Splunk. When set, log.time_unix_nano is ignored.

destination_type

Type: string
Related schema element: meta.destination.type

Type of the destination, for example, `splunkHEC`.

edge_connector_label_labels

Type: object
Related schema element: meta.edge.connector.labels

Labels set on the edge connector that sent the message.

edge_connector_name

Type: string
Related schema element: meta.edge.connector.name

The name of the edge connector that sent the message.

edge_connector_rule_id

Type: string
Related schema element: meta.edge.connector.rule_id

The ID of the owner ConnectorRule resource in Axoflow that created the edge connector.

edge_connector_type

Type: string
Related schema element: meta.edge.connector.type

The type of the edge connector that sent the message.

edge_flow_name

Type: string
Related schema element: meta.edge.flow.name

The name of the edge forwarding rule that sent the message.

flow

Type: string
Related schema element: meta.flow

Name of the flow processing the message.

host_label_labels

Type: object
Related schema element: meta.host.labels

The labels set in the inventory for the host the message originates from. Note that if the host is sending data to an AxoRouter connector that doesn't perform automatic classification, then changing the product and vendor labels can affect the final metadata in the destination, for example, the sourcetype assigned to the data in Splunk.

host_name

Type: string
Related schema element: meta.host.name

The name of the host the message originates from (based on the inventory).

host_candidate_id

Type: string
Related schema element: meta.host_candidate.id

Device ID found in the message.

host_candidate_ip

Type: string
Related schema element: meta.host_candidate.ip

IP address found in the message.

host_candidate_last_hop_name

Type: string
Related schema element: meta.host_candidate.last_hop_name

Host field found in the message's envelope (which can be either the subject of the message, or the name of the host that forwarded it).

host_candidate_name

Type: string
Related schema element: meta.host_candidate.name

Host name found in the message.

kubernetes_container

Type: string
Related schema element: meta.kubernetes.container

Kubernetes container name found in the message metadata.

kubernetes_namespace

Type: string
Related schema element: meta.kubernetes.namespace

Name of the Kubernetes namespace found in the message metadata.

product

Type: string
Related schema element: meta.product

The product name of the appliance, application, or service that generated the message.

axo_host_label_labels

Type: object
Related schema element: meta.router.labels

Labels of the AxoRouter instance that processed the message.

axo_host_name

Type: string
Related schema element: meta.router.name

The hostname of the AxoRouter instance that processed the message.

service

Type: string
Related schema element: meta.service.name

Name of the service that generated the message. For syslog messages, that's usually the value of the PROGRAM field.

vendor

Type: string
Related schema element: meta.vendor

The vendor of the appliance, application, or service that generated the message.

18.7 Storage schema

The fields that AxoStore uses to store your data, which you can use in search queries.

The following list shows the schema used when storing data in AxoStore. You can use these fields when searching on the Storage page.

log_attributes

Type: object
Related schema element: log.attributes

Additional attributes that describe the specific event occurrence. Every attribute key must be unique.

body

Type: string
Related schema element: log.body

The body of the log record, which can contain strings and structured data composed of arrays and maps of other values.

observed_time

Type: datetime
Related schema element: log.observed_time_unix_nano

Time when the event was observed by the data pipeline, in UNIX Epoch time (nanoseconds elapsed since 00:00:00 UTC on 1 January 1970). A value of `0` indicates unknown or missing timestamp.

- For events that originate in OpenTelemetry, this timestamp is typically set at the generation time and is equal to `time_unix_nano`.
- For events originating externally and collected by an Axoflow agent or an AxoRouter, this is the time when the Axoflow pipeline observed the event.

severity_number

Type: number
Related schema element: log.severity_number

Numerical value of the severity, normalized to values described in Log Data Model.

severity_text

Type: string
Related schema element: log.severity_text

The severity as a string (log level). The original string representation as described at the source. For the numerical to string mapping, see log.severity_number.

timestamp

Type: datetime
Related schema element: log.time_unix_nano

The time when the event occurred in UNIX Epoch time (nanoseconds elapsed since 00:00:00 UTC on 1 January 1970). A value of `0` indicates unknown or missing timestamp.

meta_raw

Type: object
Related schema element: meta

Metadata about a specific message record, for example, a log message.

connection

Type: object
Related schema element: meta.connection

Information about the network connection that transmitted the message.

host_labels

Type: object
Related schema element: meta.host.labels

The labels set in the inventory for the host the message originates from. Note that if the host is sending data to an AxoRouter connector that doesn't perform automatic classification, then changing the product and vendor labels can affect the final metadata in the destination, for example, the sourcetype assigned to the data in Splunk.

host_name

Type: string
Related schema element: meta.host.name

The name of the host the message originates from (based on the inventory).

host_candidate

Type: object
Related schema element: meta.host_candidate

product

Type: string
Related schema element: meta.product

The product name of the appliance, application, or service that generated the message.

axo_host_labels

Type: object
Related schema element: meta.router.labels

Labels of the AxoRouter instance that processed the message.

axo_host_name

Type: string
Related schema element: meta.router.name

The hostname of the AxoRouter instance that processed the message.

service

Type: string
Related schema element: meta.service.name

Name of the service that generated the message. For syslog messages, that's usually the value of the PROGRAM field.

vendor

Type: string
Related schema element: meta.vendor

The vendor of the appliance, application, or service that generated the message.

resource_attributes

Type: object
Related schema element: resource.attributes

Attributes that describe the resource. Every attribute key must be unique.

scope_attributes

Type: object
Related schema element: scope.attributes

Attributes that describe the log scope. Every attribute key must be unique.

scope_name

Type: string
Related schema element: scope.name

Name of the log scope

scope_version

Type: string
Related schema element: scope.version

Version of the log scope

19 Legal notice

Copyright, trademark, and third-party license information for Axoflow products.

Copyright of Axoflow Inc. All rights reserved. The Axoflow, AxoRouter, AxoLake, AxoLocker , AxoStore, and AxoSyslog trademarks and logos are trademarks of Axoflow Inc.

syslog-ng™ is a trademark of One Identity LLC.

All other trademarks are property of their respective owners.

For a list of 3rd-party components and their associated copyright and license information, see the **About** page in AxoConsole.