

NMOP  
Internet-Draft  
Intended status: Experimental  
Expires: 9 November 2025

V. Riccobene  
Huawei  
T. Graf  
W. Du  
Swisscom  
A. Huang Feng  
INSA-Lyon  
8 May 2025

An Experiment: Network Anomaly Lifecycle  
draft-ietf-nmop-network-anomaly-lifecycle-03

## Abstract

Network Anomaly Detection is the act of detecting problems in the network. Accurately detect problems is very challenging for network operators in production networks. Good results require a lot of expertise and knowledge around both the implied network technologies and the connectivity services provided to customers, apart from a proper monitoring infrastructure. In order to facilitate network anomaly detection, novel techniques are being introduced, including programmatical, rule-based and AI-based, with the promise of improving scalability and the hope to keep a high detection accuracy. To guarantee acceptable results, the process needs to be properly designed, adopting well-defined stages to accurately collect evidence of anomalies, validate their relevancy and improve the detection systems over time, iteratively.

This document describes a well-defined approach on managing the lifecycle process of a network anomaly detection system, spanning across the recording of its output and its iterative refinement, in order to facilitate network engineers to interact with the network anomaly detection system, enable the "human-in-the-loop" paradigm and refine the detection abilities over time. The major contributions of this document are: the definition of three key stages of the lifecycle process, the definition of a state machine for each anomaly annotation on the system and the definition of YANG data models describing a comprehensive format for the anomaly labels, allowing a well-structured exchange of those between all the interested actors.

## Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the Operations and Management Area Working Group Working Group mailing list ([nmop@ietf.org](mailto:nmop@ietf.org)), which is archived at <https://mailarchive.ietf.org/arch/browse/nmop/>.

Source for this draft and an issue tracker can be found at <https://github.com/network-analytics/draft-netana-nmop-network-anomaly-lifecycle/>.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 9 November 2025.

## Copyright Notice

Copyright (c) 2025 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Status of this document . . . . .	4
3. Terminology . . . . .	4
4. Defining Desired States . . . . .	5
5. Lifecycle of a Network Anomaly . . . . .	6
5.1. Network Anomaly Detection . . . . .	7

5.2. Network Anomaly Validation . . . . .	8
5.3. Network Anomaly Refinement . . . . .	8
6. Introducing a Label Store for Network Anomaly labels . . . . .	9
7. Network Anomaly State Machine . . . . .	9
8. Network Anomaly Lifecycle Data Model . . . . .	10
8.1. Overview of the Data Model for the Relevant State and all the related entities . . . . .	12
8.2. YANG Module . . . . .	13
9. Implementation status . . . . .	23
9.1. Antagonist . . . . .	23
10. Security Considerations . . . . .	23
11. Acknowledgements . . . . .	24
12. References . . . . .	24
12.1. Normative References . . . . .	24
12.2. Informative References . . . . .	26
Authors' Addresses . . . . .	27

## 1. Introduction

The main objective of a network anomaly detection system is to identify Relevant States of the network as those are States that could lead to problems or might be clear indications of Problem already happening.

An architecture for network anomaly detection is defined in [I-D.ietf-nmop-network-anomaly-architecture].

It is still remarkably difficult to gain a full understanding and a complete perspective of "if" and "how" a Relevant State is actually an indication of a Problem or it is just unexpected, but has no impact on services and end users. Providers of solutions for network anomaly detection should aim at increasing accuracy, by minimizing false positives and false negatives. Moreover, the behaviour of the network naturally changes over time. When more connectivity services are deployed, more customers are on-boarded to the network, network devices are upgraded or replaced, and therefore, it is almost impossible to identify anomaly detection techniques that can keep working accurately over time, without changing the detection criterias (or methodologies) over time.

This opens up to the necessity of further validating notified Relevant States to check if a detected symptom is actually impacting connectivity services: this might require different actors (both human and algorithmic) to act together during the process and refine their understanding across the network anomaly lifecycle.

Finally, once validation has happened, this might lead to refinements to the logic that is used by the detection, so that this process can improve the detection accuracy over time.

Performing network anomaly detection is a process that requires a continuous learning and continuous improvement. Relevant States are detected by aggregating and understanding Symptoms, then validated, confirming that Symptoms actually impacted connectivity services impacting and eventually need to be further analyzed by performing postmortem analysis to identify any potential adjustment to improve the detection capability. Each of these steps represents an opportunity to learn and refine the process, and since implementations of these steps might also be provided by different parties and/or products, this document also contributes a formal data model to capture and exchange Symptom information across the lifecycle.

The adjustment of the detection process can happen after problems are solved and well understood, so the learning can be transferred into the detection system for future faster detections. For this reason, the network anomaly Detection lifecycle mainly relates to the post-mortem stage of the network anomaly detection process, as described in [I-D.ietf-nmop-network-anomaly-architecture].

## 2. Status of this document

This document is experimental. The main goal of this document is to propose an iterative lifecycle process to network anomaly detection by proposing a data model for metadata to be addressed at different lifecycle stages.

The experiment consists of verifying whether the approach is usable in real use case scenarios to support proper refinement and adjustments of network anomaly detection algorithms. The experiment can be deemed successful if validated at least with an open-source implementation successfully applied with real networks.

## 3. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document makes use of the terms defined in [I-D.ietf-nmop-terminology].

- \* State
- \* Problem
- \* Event
- \* Alarm
- \* Symptom
- \* Relevance
- \* Network Anomaly

The following terms are used as defined in [RFC9417].

- \* Metric
- \* Intent

The following terms are defined in this document.

- \* Annotator: Is a human or an algorithm which produces metadata by describing anomalies with Symptoms.
- \* False Positive: Is a detected anomaly which has been identified during the postmortem to be not anomalous.
- \* False Negative: Is anomalous but has not been identified by the anomaly detection system.
- \* Relevant State: Is a State that has Relevance for network operators, as, for instance, those are States that could lead to Problems or might be clear indications of Problem already happening.

#### 4. Defining Desired States

The Problem, as defined in Section 3, provide the scope for what to be looking for when detecting network anomalies. Concepts like "desirable State" and "required State" are introduced in this document. This poses the attention on a significant Problem that network operators have to face: the definition of what is to be considered "desirable" or "undesirable". It is not always easy to detect if a network is operating in an undesired State at a given point in time. To approach this, network operators can rely on different methodologies, more or less deterministic and more or less sensitive: on the one side, the definition of intents (including

Service Level Objectives and Service Level Agreements) which approaches the Problem top-down; on the other side, the definition of Symptoms, by mean of solutions like SAIN [RFC9417], [RFC9418] and [I-D.ietf-nmop-network-anomaly-architecture], which approaches the Problem bottom-up. At the center of these approaches, there are the so-called Symptoms, explaining what is not working as expected in the network, sometimes also providing hints towards issues and their causes.

One of the more deterministic approaches is to rely on Symptoms based on measurable service-based KPIs, for example, by using Service Level Indicators, Objectives and Agreements [RFC9543]. This is the case when rules on SLOs and SLIs are manually defined once and the used afterwards for detection at runtime.

However, defining SLOs in a "static way" can bring some challenges as well, related to the dynamic nature of networks and services.

Alternative methodologies rely on a more "relaxed" approach to detect symptoms and their impact to services as a way to generate analytical data out of operational data. For instance:

SAIN introduces the definition and exposure of Symptoms as a mechanism for detecting those concerning behaviors in a more deterministic way. Moreover, the concept of "impact score" has been introduced by SAIN, to indicate what is the expected degree of impact that a given Symptom will have on the services relying on the related subservice to which the Symptom is attached.

Daisy introduces the concept of "concern score" to indicate what is the degree of concern that a given Symptom could cause a degradation for a connectivity service.

In general, defining boundaries between desirable vs. undesirable in an accurate fashion requires continuous iterations and improvements coming from all the stages of the network anomaly detection lifecycle, by which network engineers can transfer what they learn through the process into new Symptom definitions and, ultimately, into refinements of the detection algorithms.

## 5. Lifecycle of a Network Anomaly

The lifecycle of a network anomaly can be articulated in three phases, structured as a loop: Detection, Validation, Refinement.

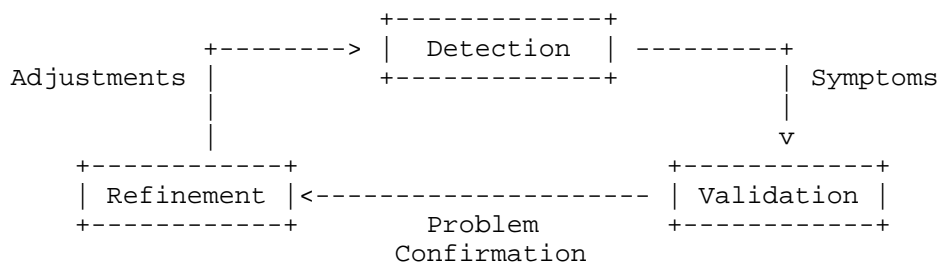


Figure 1: Anomaly Detection Refinement Lifecycle

Each of these phases can either be performed by a network expert or an algorithm or complementing each other.

The network anomaly metadata is generated by an annotator, which can be either a human expert or an algorithm. The annotator can produce the metadata for a network anomaly, for each stage of the cycle and even multiple versions for the same stage. In each version of the network anomaly metadata, the annotator indicates the list of Symptoms that are part of the network anomaly taken into account. The iterative process is about the identification of the right set of Symptoms.

### 5.1. Network Anomaly Detection

The Network Anomaly Detection stage is about the continuous monitoring of the network through Network Telemetry [RFC9232] and the identification of Symptoms. One of the main requirements that operator have on network anomaly detection systems is the high accuracy. This means having a small number of false negatives, i.e. Symptoms causing connectivity service impact are not missed, and low amount of false positives, i.e. Symptoms that are actually innocuous are not picked up.

As the detection stage is becoming more and more automated for production networks, the identified Symptoms might point towards three potential kinds of behaviors:

- i. those that are surely corresponding to an impact on connectivity services, (e.g. the breach of an SLO),
- ii. those that will cause Problems in the future (e.g. rising trends on a timeseries metric hitting towards saturation),
- iii. those or which the impact to connectivity services cannot be confirmed (e.g. sudden increase/decrease of timeseries metrics, anomalous amounts of log entries, etc.).

The first category requires immediate intervention (a.k.a. the problem is "confirmed"), the second one provides pointers towards early signs of an problem potentially happening in the near future (a.k.a. the problem is "forecasted"), and the third one requires some analysis to confirm if the detected Symptom requires any attention or immediate intervention (a.k.a. the problem is "potential"). As part of the iterative improvement required in this stage, one that is very relevant is the gradual conversion of the third category into one of the first two, which would make the network anomaly detection system more deterministic. The main objective is to reduce uncertainty around the raised alarms by refining the detection algorithms. This can be achieved by either generating new Symptom definitions, adjusting the weights of automated algorithms or other similar approaches.

## 5.2. Network Anomaly Validation

The key objective for the validation stage is clearly to decide if the detected Symptoms are signaling a real problem (a.k.a. requires action) or if they are to be treated as false positives (a.k.a. suppressing the alarm). For those Symptoms surely having impact on connectivity services, 100% confidence on the fact that a network problem is happening can be assumed. For the other two categories, "forecasted" and "potential", further analysis and validation is required.

## 5.3. Network Anomaly Refinement

After validation of a problem, the service provider performs troubleshooting and resolution of the problem. Although the network might be back in a desired State at this point, network operators can perform detailed postmortem analysis of Problems with the objective to identify useful adjustments to the prevention and detection mechanisms (for instance improving or extending the definition of SLIs and SLOs, refining concern/impact scores, etc.), and improving the accuracy of the validation stage (e.g. automating parts of the validation, implementing automated root cause analysis and automation for remediation actions). In this stage of the lifecycle it is assumed that the problem is under analysis.

After the adjustments are performed to the network anomaly detection methods, the cycle starts again, by "replaying" the network anomaly and checking if there is any measurable improvement in the ability to detect Problems by using the updated method.

## 6. Introducing a Label Store for Network Anomaly labels

The information that is produced at each stage needs to be persisted and retrieved to perform the network anomaly lifecycle.

The lifecycle begins with the detector notifying anomalies to the "Alarm and Problem Management System" and to the "Post-mortem System" according to (see [I-D.ietf-nmop-network-anomaly-architecture]). In this case the Post-mortem system is identified as the Label Store. Once the notification arrives to the Label Store, the anomaly label is persisted. In the following stages (i.e. validation and refinement), the information about the labels are retrieved, reviewed, modified and persisted again, generating every time a new version of the same annotation, or tagging the annotation as irrelevant, if it would be necessary to remove it.

In the following sections, the following are defined:

- \* a state machine for a label
- \* a YANG data model for the notification sent by the Detector to the Label Store
- \* a YANG data model to define the interrogation (and retrieval) of the persisted labels from the label store.

## 7. Network Anomaly State Machine

In the context of this document, from a network anomaly detection point of view a Problem is defined as a collection of interrelated Symptoms, as specified in [I-D.netana-nmop-network-anomaly-semantics].

The understanding of a Problem can change over time. Moreover, multiple actors are involved in the process of refining this understanding in the different phases.

From this perspective, a problem can be refined according to the following States (Figure 2).

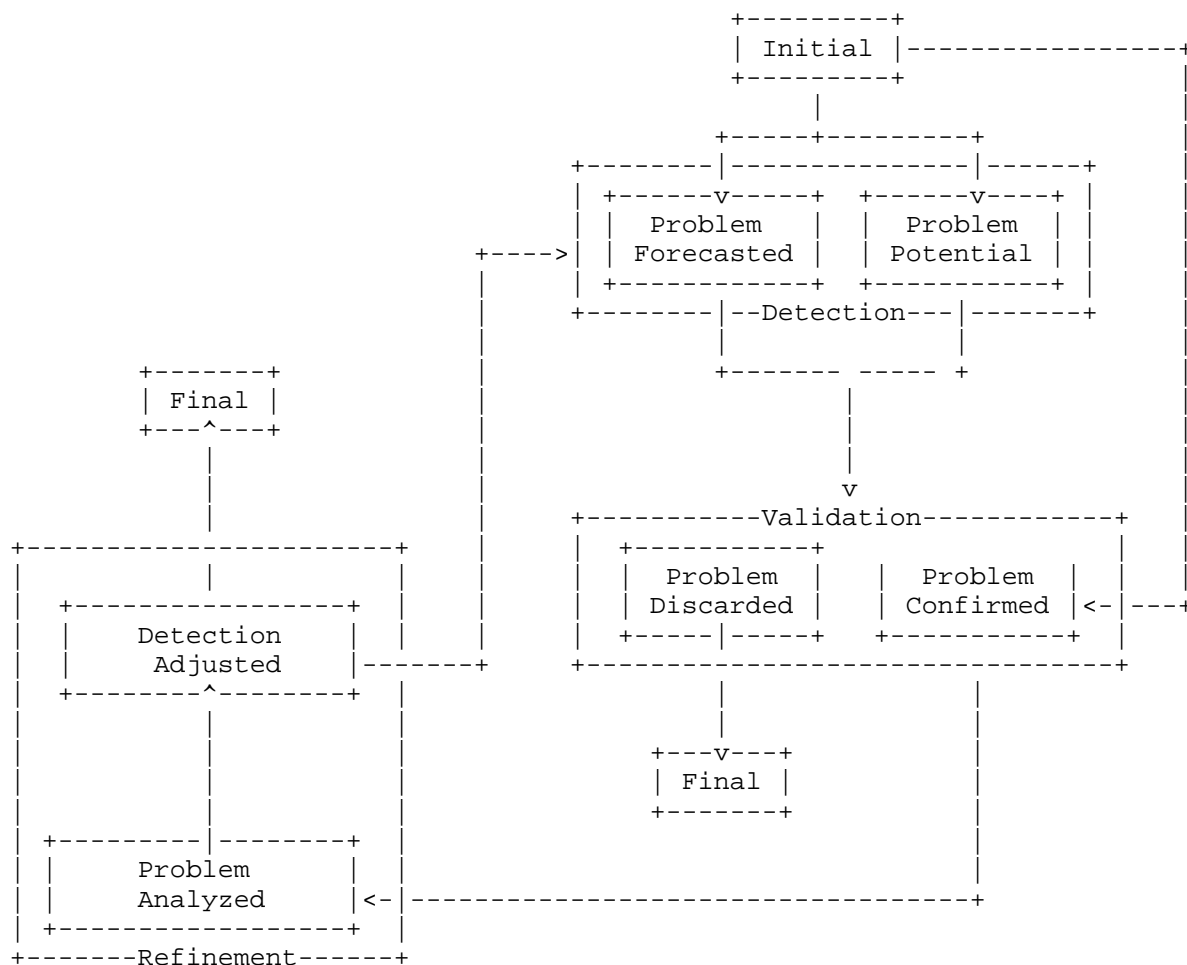


Figure 2: Network Anomaly State Machine

The knowledge gained at each stage is encoded as a list of anomaly labels that can be stored on a Label Store (see Section 9.1 for a reference implementation).

## 8. Network Anomaly Lifecycle Data Model

The data model provides support for "human-in-the-loop", allowing for network experts to validate and adjust network anomaly labels and detection systems. An example of human-in-the-loop has been demonstrated with Antagonist [Antagonist], by building a User Interface that interacts with an API based on this data model.

The base for the modules is the "ietf-relevant-state" data model. Relevant State is at the root of the data model, with its parameters (ID, description, start-time, end-time) and a collection of anomalies. This allows the relevant state to be considered as a container of anomalies.

Each anomaly is characterized by some intrinsic fields (such as id, version, state, description, start-time, end-time, confidence score and pattern) Particularly the confidence score is a measure of how confident was the detector in considering the given anomaly as an anomalous behaviour.

Each anomaly also include the symptom and the service container. These containers are placeholders to represent the information about the symptom (what is exactly happening as anomalous behaviour) and the connectivity service (what entity is affected by the anomaly). In particular, for what concerns the symptom, a concern score is defined as necessary field, which has the meaning of expressing how much the anomaly is impacting connectivity services.

Concern and Confidence scores are used to express two very different concepts, and it is important to make a clear distinction between them:

- \* the Confidence score tends to be higher when symptoms are more abnormal, for instance if some given values are very rare in the data or if they substantially deviate from expected behaviours. In those cases, detectors are more sure about the anomaly and this is reflected into higher confidence scores.
- \* the Concern score tends to be higher when the symptom is likely having a higher impact on connectivity services.

Although the two concepts can be somehow intertwined, in the scope of this document a clear way on how to use them it is not provided, as this has been shown to be dependent on the use case and on the user of the system.

In case additional information related to the symptom and to the service need to be provided, augmentation would be the appropriate intended mechanism to do so. An example of this is provided in [I-D.netana-nmop-network-anomaly-semantics], where an augmentation of both symptom and service is provided for the specific case of anomaly labels related to connectivity services.

Also a list of various actors that can be involved in the process is presented as following:

In the detection stage: the detectors can be Network Engineers and/or Automatic detectors (including Rule-based detectors and ML-based detectors)

In the validation stage: the validators can be Network Engineers manually validating the labels

In the refinement stage: the refiners can be Data Scientists and/or Automatic Refiners (including systems that automatically refine the detection systems, based on the validated labels).

The data model defines a Relevant State container and a Relevant State notification: the notification is primarily used by the Network Anomaly Detector, to notify the "Alarm and Problem Management System" and the "Post-mortem System" (see [I-D.ietf-nmop-network-anomaly-architecture]); the container instead is used inside the Post-mortem system to exchange anomaly detection labels between the anomaly detection stages defined above (validation, refinement, detection).

#### 8.1. Overview of the Data Model for the Relevant State and all the related entities

The "ietf-relevant-state" module defines a Relevant State container to report associated anomalies and a "relevant-state-notification" used for exchanging the relevant State information. The following YANG tree diagram [RFC8340] shows the "ietf-relevant-state" model.

```
module: ietf-relevant-state
  +--rw relevant-state
    +--rw id                               yang:uuid
    +--rw uri?                             inet:uri
    +--rw description?                     string
    +--rw start-time                       yang:date-and-time
    +--rw end-time?                        yang:date-and-time
    +--rw strategy?                        string
    +--rw confidence-score?                score
    +--rw concern-score                    score
    +--rw (service)?
    +--rw anomaly* [id revision]
      +--rw id                             yang:uuid
      +--rw revision                       yang:counter32
      +--rw uri?                           inet:uri
      +--rw state                           identityref
      +--rw description?                   string
      +--rw start-time                     yang:date-and-time
      +--rw end-time?                      yang:date-and-time
      +--rw confidence-score?              score
```

```

+--rw pattern?                identityref
+--rw annotator
|   +--rw id?                 yang:uuid
|   +--rw name                string
|   +--rw version?           string
|   +--rw annotator-type?    enumeration
+--rw symptom!
    +--rw id                  yang:uuid
    +--rw concern-score      score

```

#### notifications:

```

+---n relevant-state-notification
+--ro publisher
|   +--ro id?                 yang:uuid
|   +--ro name                string
|   +--ro version?           string
+--ro id                     yang:uuid
+--ro uri?                   inet:uri
+--ro description?           string
+--ro start-time              yang:date-and-time
+--ro end-time?               yang:date-and-time
+--ro strategy?               string
+--ro confidence-score?      score
+--ro concern-score          score
+--ro (service)?
+--ro anomaly* [id revision]
    +--ro id                  yang:uuid
    +--ro revision             yang:counter32
    +--ro uri?                 inet:uri
    +--ro state                identityref
    +--ro description?         string
    +--ro start-time           yang:date-and-time
    +--ro end-time?            yang:date-and-time
    +--ro confidence-score?    score
    +--ro pattern?             identityref
    +--ro annotator
        +--ro id?              yang:uuid
        +--ro name              string
        +--ro version?          string
        +--ro annotator-type?  enumeration
    +--ro symptom!
        +--ro id                yang:uuid
        +--ro concern-score     score

```

## 8.2. YANG Module

```
<CODE BEGINS> file "ietf-relevant-state@2025-03-23.yang"
module iETF-relevant-state {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-relevant-state";
  prefix rsN;

  import iETF-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types";
  }
  import iETF-inet-types {
    prefix inet;
    reference
      "RFC 6991: Common YANG Data Types";
  }

  organization
    "IETF NMOP Working Group";
  contact
    "WG Web:  <https://datatracker.ietf.org/wg/nmop/>
    WG List:  <mailto:nmop@ietf.org>

    Editor:   Vincenzo Riccobene
              <mailto:vincenzo.riccobene@huawei-partners.com>
              Thomas Graf
              <mailto:thomas.graf@swisscom.com>
              Wanting Du
              <mailto:wanting.du@swisscom.com>
              Alex Huang Feng
              <mailto:alex.huang-feng@insa-lyon.fr>";
  description
    "This module defines the relevant-state container and
    notifications to be used by a network anomaly detection
    system. The defined objects can be used to augment
    operational network collected observability data and
    analytical problem data equally. Describing the relevant-state
    of observed symptoms.

    Copyright (c) 2025 IETF Trust and the persons
    identified as authors of the code.  All rights reserved.

    Redistribution and use in source and binary forms, with or
    without modification, is permitted pursuant to, and subject
    to the license terms contained in, the Revised BSD License
    set forth in Section 4.c of the IETF Trust's Legal Provisions
    Relating to IETF Documents
    (https://trustee.ietf.org/license-info).
```

All revisions of IETF and IANA published modules can be found at the YANG Parameters registry (<https://www.iana.org/assignments/yang-parameters>).

This version of this YANG module is part of RFC XXXX; see the RFC itself for full legal notices."

```
revision 2025-04-25 {
  description
    "Initial version";
  reference
    "RFC XXX: Semantic Metadata Annotation for Network Anomaly
    Detection";
}

typedef score {
  type uint8 {
    range "0..100";
  }
  description
    "Number that indicates a score between 0 and 100.";
}

identity network-anomaly-state {
  description
    "Base identity for representing the state of the network
    anomaly.";
  reference
    "Section 6 in draft-ietf-nmop-network-anomaly-lifecycle.";
}

identity detection {
  base network-anomaly-state;
  description
    "A problem reached detection state.";
  reference
    "Section 6.1 in draft-ietf-nmop-network-anomaly-lifecycle.";
}

identity validation {
  base network-anomaly-state;
  description
    "A problem reached validation state.";
  reference
    "Section 6.2 in draft-ietf-nmop-network-anomaly-lifecycle.";
}

identity refinement {
```

```
base network-anomaly-state;
description
  "A problem reached refinement state.";
reference
  "Section 6.3 in draft-ietf-nmop-network-anomaly-lifecycle.";
}

identity problem-forecasted {
  base detection;
  description
    "A problem has been forecasted, as it is expected that
    the indicated list of symptoms will impact a service
    in the near future.";
}

identity problem-potential {
  base detection;
  description
    "A problem has been detected with a confidence
    lower than 100%. In order to confirm that this set of
    symptoms are generating service impact, it requires further
    validation.";
}

identity problem-confirmed {
  base validation;
  description
    "After validation, the problem has been confirmed.";
}

identity discarded {
  base validation;
  description
    "After validation, the network anomaly has been
    discarded, as there is no evindence that it is causing a
    problem.";
}

identity analyzed {
  base refinement;
  description
    "The anomaly detection went through analysis to identify
    potential ways to further improve the detection process in
    for future anomalies.";
}

identity adjusted {
  base refinement;
```

```
    description
    "The network anomaly has been solved and analysed.
    No further action is required.";
}

identity pattern {
    description
    "Pattern identified by the Detector.";
}

identity drop {
    base pattern;
    description
    "Drop of the value.";
}

identity spike {
    base pattern;
    description
    "Spike of the value.";
}

identity mean-shift {
    base pattern;
    description
    "Shift of the mean of the value.";
}

identity seasonality-shift {
    base pattern;
    description
    "Shift of the seasonality of the value.";
}

identity trend {
    base pattern;
    description
    "Trend exhibited by the value.";
}

identity other {
    base pattern;
    description
    "Any other type of pattern.";
}

grouping relevant-state-grouping {
    description
```

```
    "Relevant State is a state that could lead to
    problems or might be clear indications of problem
    already happening.";
reference
    "Figure 4 in draft-ietf-nmop-terminology.";
leaf id {
    type yang:uuid;
    mandatory true;
    description
        "Unique ID of the relevant state. It is unique
        in the scope of the Label Store.";
}
leaf uri {
    type inet:uri;
    description
        "URI to viusalize the analytical metrics of the
        relevant-state.";
}
leaf description {
    type string;
    description
        "Textual description of the fault.";
}
leaf start-time {
    type yang:date-and-time;
    mandatory true;
    description
        "Date and time indicating the beginning of the problem.";
}
leaf end-time {
    type yang:date-and-time;
    description
        "Date and time indicating the end of the problem.";
}
leaf strategy {
    type string;
    mandatory false;
    description
        "Captures one approach to look at the data (as a human operator
        does) to observe if an abnormal situation is arising.";
    reference
        "Section 3.5.3 in
        draft-ietf-nmop-network-anomaly-architecture.";
}
leaf confidence-score {
    type score;
    description
        "Score indicating how confident were the detectors
```

```
        in relation to the overall relevant state.";
    }
    leaf concern-score {
        type score;
        mandatory true;
        description
            "Score indicating the degree of concern in
             relation to the overall relevant state.";
    }
    choice service {
        description
            "Indication of the service that is affected
             (or potentially affected) by the relevant state";
    }
}

grouping annotator-grouping {
    description
        "Annotator represents the entity that produced the
         annotation (it is either a human or an algorithm).";
    leaf id {
        type yang:uuid;
        description
            "Unique ID of the annotator (either user or algorithm).";
    }
    leaf name {
        type string;
        mandatory true;
        description
            "Name of the annotator (either user or algorithm).";
    }
    leaf version {
        type string;
        mandatory false;
        description
            "Version of the annotator.";
    }
    leaf annotator-type {
        type enumeration {
            enum human {
                description
                    "This option is used if a human provided the label.";
            }
            enum algorithm {
                description
                    "This option is used if a algorithm or software
                     provided the label.";
            }
        }
    }
}
```

```
    }
    description
      "An annotator can be either a human user or a
       programmatic entity, such as an algorithm.";
  }
}

grouping anomaly-grouping {
  description
    "List of anomalies that are part of the relevant state";
  list anomaly {
    key "id revision";
    description
      "List of Anomaly instances.";
    leaf id {
      type yang:uuid;
      description
        "Unique identifier of the anomaly.";
    }
    leaf revision {
      type yang:counter32;
      description
        "Revision of the anomaly metadata object.
         It allows multiple revisions of the metadata to be
         generated in order to support the definition of
         multiple problem objects from the same source to
         facilitate improvements overtime.";
    }
    leaf uri {
      type inet:uri;
      description
        "URI to viusalize the analytical metrics of the
         anomaly.";
    }
    leaf state {
      type identityref {
        base network-anomaly-state;
      }
      mandatory true;
      description
        "State of the anomaly.";
    }
    leaf description {
      type string;
      description
        "Textual description of the anomaly.";
    }
    leaf start-time {
```

```
type yang:date-and-time;
mandatory true;
description
    "Date and time indicating the beginning of the anomaly.
    A detection system will always set a start time,
    as it represents the moment in time from which the
    behaviour of the monitored system is considered
    to be anomalous with respect its expected behaviour.";
}
leaf end-time {
    type yang:date-and-time;
    description
        "Date and time indicating the end of the anomaly.
        This field is indicated as non mandatory, as it could
        be the case that the anomaly is still happening at the
        time of generation of the label.";
}
leaf confidence-score {
    type score;
    description
        "Score indicating how confident was the detector
        while considering the given anomaly as part of the
        relevant event.";
}
leaf pattern {
    type identityref {
        base pattern;
    }
    description
        "Pattern describes the type of pattern that was
        detected by the annotator (e.g. spike, drop,
        mean-shift, etc.).";
}
container annotator {
    description
        "Annotator represents the entity that produced the
        annotation.";
    uses annotator-grouping;
}
container symptom {
    presence "It specifies the symptom for the anomaly";
    description
        "An observable Characteristic, State, Event, or
        Condition considered as an indication of a Problem
        or potential Problem.";
    leaf id {
        type yang:uuid;
        mandatory true;
    }
}
```

```
        description
            "Unique identifier of the symptom type.";
    }
    leaf concern-score {
        type score;
        mandatory true;
        description
            "Score indicating the degree of concern in
             relation to the specific symptom. Each
             symptom will carry a certain degree of
             concern that is specific to the symptom.";
    }
}
}
}

notification relevant-state-notification {
    description
        "Notification of a relevant state that can be sent by the
         anomaly detection system to the postmortem management
         system or to the incident management system.";
    container publisher {
        description
            "Publisher represents the entity that produced the
             relevant-state.";
        leaf id {
            type yang:uuid;
            description
                "Unique ID of the publisher.";
        }
        leaf name {
            type string;
            mandatory true;
            description
                "Name of the publisher.";
        }
        leaf version {
            type string;
            mandatory false;
            description
                "Version of the publisher.";
        }
    }
    uses relevant-state-grouping;
    uses anomaly-grouping;
}

container relevant-state {
```

```
    description
      "A Relevant State is a state that have relevancy
      for network operators, as those are states that could lead
      to problems or might be clear indications of problem already
      happening.";
    uses relevant-state-grouping;
    uses anomaly-grouping;
  }
}
<CODE ENDS>
```

## 9. Implementation status

This section provides pointers to existing open source implementations of this draft. Note to the RFC-editor: Please remove this before publishing.

### 9.1. Antagonist

An open-source implementation for this draft is called AnTagOnIst (Anomaly Tagging On hIstorical data), and it has been implemented in order to validate the application of the YANG model defined in this draft. Antagonist provides visual support for two important use cases in the scope of this document:

- \* the generation of a ground truth in relation to symptoms and Problems in timeseries data
- \* the visual validation of results produced by automated network anomaly detection tools.

The open-source code can be found here: [Antagonist]

As part of the experiment that was conducted with AnTagOnIst, Some main Use Case scenarios have been validated so far:

Exposure of a GUI for human validation of the labels.

Integration with Rule Based anomaly detection systems. In particular the integration with SAIN and Cosmos Bright Lights is ongoing.

Integration with ML-based detection systems.

## 10. Security Considerations

This section is modeled after the template described in Section 3.7 of [I-D.ietf-netmod-rfc8407bis].

The "ietf-network-anomaly-symptom-cbl" and "ietf-network-anomaly-service-topology" YANG modules defines two data models that are designed to be accessed via YANG-based management protocols, such as NETCONF [RFC6141] and RESTCONF [RFC8040]. These protocols have to use a secure transport layer (e.g., SSH [RFC4252], TLS [RFC8446], and QUIC [RFC9000]) and have to use mutual authentication.

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., "config true", which is the default). All writable data nodes are likely to be reasonably sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) and delete operations to these data nodes without proper protection or authentication can have a negative effect on network operations. The following subtrees and data nodes have particular sensitivities/vulnerabilities:

"There are no particularly sensitive writable data nodes."

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. Specifically, the following subtrees and data nodes have particular sensitivities/vulnerabilities:

"There are no particularly sensitive readable data nodes."

## 11. Acknowledgements

The authors would like to thank Antonio Roberto for his contribution to the ideas in this draft and Mohamed Boucadair and Adrian Farrel for his review and valuable comments.

## 12. References

### 12.1. Normative References

[Antagonist]

Riccobene, V., Du, W., Graf, T., and H. Huang Feng,  
"Antagonist: Anomaly tagging on historical data",  
<<https://github.com/vriccobene/antagonist>>.

[I-D.ietf-nmop-network-anomaly-architecture]

Graf, T., Du, W., Francois, P., and A. H. Feng, "A Framework for a Network Anomaly Detection Architecture", Work in Progress, Internet-Draft, draft-ietf-nmop-network-anomaly-architecture-03, 8 May 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-nmop-network-anomaly-architecture-03>>.

[I-D.ietf-nmop-terminology]

Davis, N., Farrel, A., Graf, T., Wu, Q., and C. Yu, "Some Key Terms for Network Fault and Problem Management", Work in Progress, Internet-Draft, draft-ietf-nmop-terminology-16, 15 April 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-nmop-terminology-16>>.

[I-D.netana-nmop-network-anomaly-semantics]

Graf, T., Du, W., Feng, A. H., Riccobene, V., and A. Roberto, "Semantic Metadata Annotation for Network Anomaly Detection", Work in Progress, Internet-Draft, draft-netana-nmop-network-anomaly-semantics-04, 3 November 2024, <<https://datatracker.ietf.org/doc/html/draft-netana-nmop-network-anomaly-semantics-04>>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

[RFC4252] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Authentication Protocol", RFC 4252, DOI 10.17487/RFC4252, January 2006, <<https://www.rfc-editor.org/info/rfc4252>>.

[RFC6141] Camarillo, G., Ed., Holmberg, C., and Y. Gao, "Re-INVITE and Target-Refresh Request Handling in the Session Initiation Protocol (SIP)", RFC 6141, DOI 10.17487/RFC6141, March 2011, <<https://www.rfc-editor.org/info/rfc6141>>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <<https://www.rfc-editor.org/info/rfc9000>>.
- [RFC9232] Song, H., Qin, F., Martinez-Julia, P., Ciavaglia, L., and A. Wang, "Network Telemetry Framework", RFC 9232, DOI 10.17487/RFC9232, May 2022, <<https://www.rfc-editor.org/info/rfc9232>>.
- [RFC9417] Claise, B., Quilbeuf, J., Lopez, D., Voyer, D., and T. Arumugam, "Service Assurance for Intent-Based Networking Architecture", RFC 9417, DOI 10.17487/RFC9417, July 2023, <<https://www.rfc-editor.org/info/rfc9417>>.
- [RFC9418] Claise, B., Quilbeuf, J., Lucente, P., Fasano, P., and T. Arumugam, "A YANG Data Model for Service Assurance", RFC 9418, DOI 10.17487/RFC9418, July 2023, <<https://www.rfc-editor.org/info/rfc9418>>.
- [RFC9543] Farrel, A., Ed., Drake, J., Ed., Rokui, R., Homma, S., Makhijani, K., Contreras, L., and J. Tantsura, "A Framework for Network Slices in Networks Built from IETF Technologies", RFC 9543, DOI 10.17487/RFC9543, March 2024, <<https://www.rfc-editor.org/info/rfc9543>>.

## 12.2. Informative References

- [I-D.ietf-netmod-rfc8407bis] Bierman, A., Boucadair, M., and Q. Wu, "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", Work in Progress, Internet-Draft, draft-ietf-netmod-rfc8407bis-24, 18 April 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-netmod-rfc8407bis-24>>.

## Authors' Addresses

Vincenzo Riccobene  
Huawei  
Dublin  
Ireland  
Email: [vincenzo.riccobene@huawei-partners.com](mailto:vincenzo.riccobene@huawei-partners.com)

Thomas Graf  
Swisscom  
Binzring 17  
CH-8045 Zurich  
Switzerland  
Email: [thomas.graf@swisscom.com](mailto:thomas.graf@swisscom.com)

Wanting Du  
Swisscom  
Binzring 17  
CH-8045 Zurich  
Switzerland  
Email: [wanting.du@swisscom.com](mailto:wanting.du@swisscom.com)

Alex Huang Feng  
INSA-Lyon  
Lyon  
France  
Email: [alex.huang-feng@insa-lyon.fr](mailto:alex.huang-feng@insa-lyon.fr)