NMOP Working Group
Internet-Draft
Intended status: Standards Track
Expires: 9 November 2024

T. Hu
CMCC
L. M. C. Murillo
Telefonica I+D
Q. Wu
Huawei
N. Davis
Ciena
C. Feng
8 May 2024

A YANG Data Model for Network Incident Management
draft-ietf-nmop-network-incident-yang-00

Abstract

A network incident refers to an unexpected interruption of a network service, degradation of a network service quality, or sub-health of a network service. Different data sources including alarms, metrics and other anomaly information can be aggregated into few amount of network incidents by data correlation analysis and the service impact analysis.

This document defines YANG Modules for the network incident lifecycle management. The YANG modules are meant to provide a standard way to report, diagnose, and resolve network incidents for the sake of network service health and root cause analysis.

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 2024.

Table of Contents

1. Introduction
2. Conventions and Definitions
3. Sample Use Cases
   3.1. Incident-Based Trouble Tickets dispatching
   3.2. Incident Derivation from L3VPN services Unavailability
   3.3. Multi-layer Fault Demarcation
   3.4. Security events Automated Noise reduction based on Situation awareness
4. Network Incident Management Architecture
   4.1. Interworking with Alarm Management
   4.2. Interworking with SAIN
   4.3. Relationship with RFC8969
   4.4. Relationship with Trace Context
5. Functional Interface Requirements between the Client and the Server
   5.1. Incident Identification
   5.2. Incident Diagnosis
   5.3. Incident Resolution
6. Incident Data Model Concepts
   6.1. Identifying the Incident Instance
   6.2. The Incident Lifecycle
      6.2.1. Incident Instance Lifecycle
      6.2.2. Operator Incident Lifecycle
7. Incident Data Model Design
   7.1. Overview
   7.2. Incident Notifications
   7.3. Incident Acknowledge
   7.4. Incident Diagnose
   7.5. Incident Resolution
8. Network Incident Management YANG Module
9. Security Considerations
10. IANA Considerations
   10.1. The "IETF XML" Registry
   10.2. The "YANG Module Names" Registry
Acknowledgments
References
   Normative References
   Informative References
Appendix A. Changes between Revisions
Contributors
Authors' Addresses

1. Introduction

[RFC8969] defines a framework for Automating Service and Network Management with YANG to full life cycle network management. A set of YANG data models have already been developed in IETF for Network performance monitoring and fault monitoring,e.g.,A YANG [RFC7950] data model for alarm management [RFC8632] defines a standard interface for alarm management. A data model for Network and VPN Service Performance Monitoring [RFC9375] defines a standard interface for network performance management. In addition, distributed tracing mechanism defined in [W3C-Trace-Context] can also be used to analyze and debug operations, such as configuration transactions, across multiple distributed systems.

However these YANG data models for network maintenance are based on specific data source information and manage alarms and performance metrics data separately by different layers in various different management systems. In addition, the frequency and quantity of alarms and performance metrics data reported to Operating Support System (OSS) are increased dramatically (in many cases multiple orders of magnitude) with the growth of service types and complexity and greatly overwhelm OSS platforms; with existing known dependency relation between fault, alarm and events at each layer (e.g.,packet layer or optical layer), , it is possible to compress a series of alarms into fewer incidents and there are many solutions in the market today that essentially do this to some degree. However traditional solutions such as data compression are time-consuming and labor-intensive, usually rely on maintenance engineers' experience for data analysis, which result in low processing efficiency, inaccurate root cause identification and duplicated tickets. In addition, it is also difficult to assess the impact of alarms, performance metrics and other anomaly data on network services without known relation across layer of the network topology data or the relation with other network topology data.

To address these challenges, a network wide incident-centric solution is proposed to establish dependency relation with both network service and network topology at different layers , which not only can be used at specific layer in specific domain but also can be used to span across layer for multi-layer network troubleshooting. A network incident refers to an unexpected interruption of a network service, degradation of a network service quality, or sub-health of a network service [TMF724A]. Different data sources including alarms, metrics and other anomaly information can be aggregated into few amount of incidents irrespective layer by correlation analysis and the service impact analysis. For example, the protocols related to the interface fail to work properly due to Service Level Objective (SLO) violation, large amount of alarms may be reported to upper layer management system and aggregated into one or a few incidents when some network services may be affected by this incident (e.g. L3VPN services

related with the interface will become unavailable [I-D.ietf-ippm-pam]). An incident may also be raised through the analysis of some network performance metrics, for example, as described in SAIN [RFC9417] , network services can be decomposed to several sub-services, specific metrics are monitored for each sub-service, symptoms will occur if services/sub-services are unhealthy (after analyzing metrics), these symptoms may raise one incident when it causes degradation of the network services.

In addition, Artificial Intelligence (AI) and Machine Learning (ML) play a important role in the processing of large amounts of data with complex data correlations. For example, Neural Network Algorithm or Hierarchy Aggregation Algorithm can be used to replace manual alarm data correlation. Through online and offline learning, these algorithms can be continuously optimized to improve the efficiency of fault diagnosis.

This document defines a YANG data model for network incident lifecycle management, which improves troubleshooting efficiency, ensures network service quality, and improves network automation [RFC8969].

2. Conventions and Definitions

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.

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.

The following terms are defined in [RFC8632] and are not redefined here:

* alarm

The following terms are defined in this document:

Network Incident: An unexpected interruption of a network service, degradation of network service quality, or sub-health of a network service [TMF724A].

Problem: The cause of one or more incidents. The cause is not usually known when a problem record is created, and the problem management process is responsible for further investigation [TMF724A].

Incident management: Lifecycle management of incidents including incident identification, reporting, acknowledge, diagnosis, and resolution.

Incident management system: An entity which implements incident management. It include incident management server and incident management client.

Incident management server: An entity which provides some functions of incident management. For example, it can detect an incident, perform incident diagnosis, resolution and prediction,etc.

Incident management client: An entity which can manage incidents. For example, it can receive incident notifications, query the information of incidents, instruct the incident management server to diagnose, resolve, etc.

3. Sample Use Cases

3.1. Incident-Based Trouble Tickets dispatching

Traditionally, the dispatching of trouble tickets is mostly based on alarms data analysis and need to involve operators' maintenance engineers. These operators' maintenance engineers are able to monitor and detect that alarms at both end devices of specific network tunnel or at both optical layer and IP layer which are associated with the same network fault. Therefore, they can correlate these alarms to the same trouble ticket, which is in the low automation. If there are more alarms, then the human costs for network maintenance are increased accordingly.

Some operators preconfigure whitelist and adopt some coarse granularity data correlation rules for the alarm management. It seems to improve fault management automation. However, some trouble tickets could be missed if the filtering conditions are too strict. If the filtering conditions are not strict, it might end up with multiple trouble tickets being dispatched to the same network fault.

It is hard to achieve a perfect balance between the network management automation and duplicated trouble tickets under the traditional working situations. However, with the help of the network incident management, massive alarms can be aggregated into a few network incidents based on service impact analysis, the number of trouble tickets will be greatly reduced. At the same time, the efficiency of network troubleshooting can be largely improved. which address the pain point of traditional trouble ticket dispatching.

3.2. Incident Derivation from L3VPN services Unavailability

The service attachment points defined in [RFC9408] represent the network reference points where network services can be delivered to customers.

SLOs can be used to characterize the ability of a particular set of nodes to communicate according to certain measurable expectations [I-D.ietf-ippm-pam]. For example, an SLA might state that any given SLO applies to at least a certain percentage of packets, allowing for a certain level of packet loss and exceeding packet delay threshold to take place. An SLA might establish a multi-tiered SLO of end to end latency as follows:

* not to exceed 30 ms for any packet;

* not to exceed 25 ms for 99.999% of packets;

* not to exceed 20 ms for 99% of packets.

This SLA information can be bound with two or multiple service attachment point defined in [RFC9408], so that the service orchestration layer can use these interfaces to commit the delivery of a service on specific point to point service topology or point to multi-point topology. Upon specific levels of a threshold of an SLO is violated, a specific network incident, associated with,let's say L3VPN service will be derived.

3.3. Multi-layer Fault Demarcation

When a fault occurs in a network that contains both packet-layer devices and optical-layer devices, it may cause correlative faults in both layers, i.e., packet layer and optical layer. Specifically, fault propagation could be classified into three typical types. First, fault occurs at a packet-layer device will further cause fault (e.g.,WDM (wavelength division multiplexing) client fault) at an optical-layer device. Second, fault occurs at an optical-layer device will further cause fault (e.g., L3 link down) at a packet-layer device. Third, fault occurs at the inter-layer link between a

packet-layer device and an optical-layer device will further cause faults at both devices. Traditionally, multiple operation teams are needed to first analyze huge amount of alarms (triggered by the above mentioned faults) from single network layer independently, then cooperate to locate the root cause through manually analyzing multi-layer topology data and service data, thus fault demarcation becomes more complex and time-consuming in multi-layer scenario than in single-layer scenario.

With the help of network incident management, the management systems first automatically analyze root cause of the alarms at each single network layer and report corresponding incidents to the upper layer, then multi-layer management system comprehensively analyzes the topology relationship and service relationship between the root causes of both layers. The inner relationship among the alarms will be identified and finally the root cause will be located among

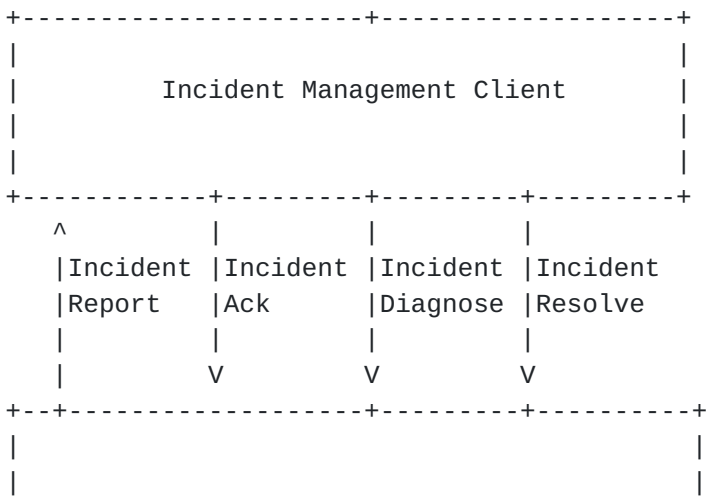
multiple layers. By cooperating with the integrated Optical time-domain reflectometer (OTDR) within the network device, we can determine the target optical exchange station before site visits. Therefore, the overall fault demarcation process is simplified and automated, the analyze result could be reported and visualized in time. In this case, operation teams only have to confirm the analyze result and dispatch site engineers to perform relative maintenance actions (e.g., splice fiber) based on the root cause.

3.4. Security events Automated Noise reduction based on Situation awareness

In the continuous data driven monitoring, tools used by the Security Operation Center (SoC) scan the network 24/7 to flag any abnormalities or suspicious activities. As the SoC adds more tools for security events detection, the volume of security events or alerts grows continually. the overwhelming number of threat alerts can cause threat fatigue. In addition, many of these alerts do not provide sufficient intelligence, context to investigate, or are false positives. False positives not only drain time and resources, but can also distract teams from real incidents.

With the help of the network incident management, BERT(Bidirectional Encoder Representations from Transformers) [BERT] classifier can be adopted to analyses the suspicious activity and understands the significance of the gathered data (through both facts and inferences) and help operation and maintenance engineers focus on handling important security events and avoid wasting resources on false alerts, e.g., automatically determine whether 10,000 network security events are real incidents and give reasonable explanations. Progressively, Llama interpreter can be used to explain the reason why such alerts are picked out and marked significant, what could be the potential security implications that exist yet remain undiscovered.

4. Network Incident Management Architecture



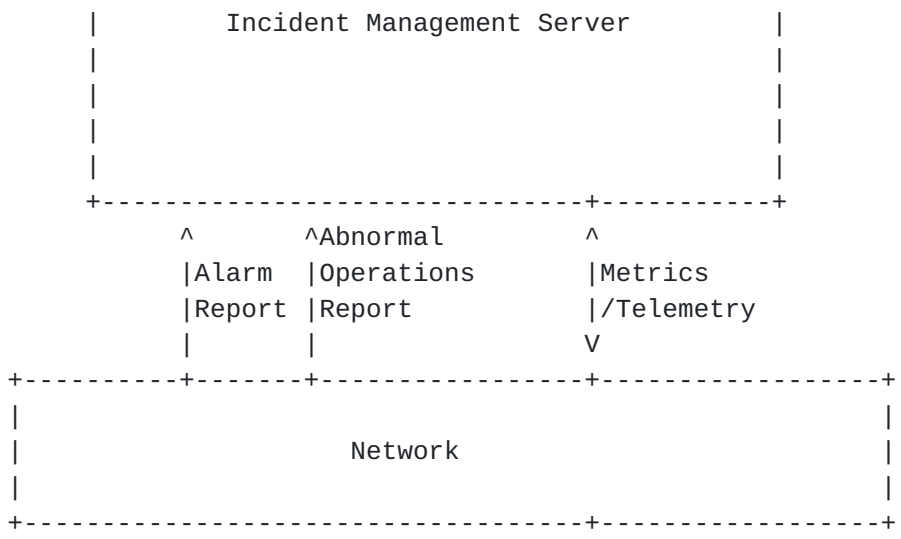


Figure 1: Network Incident Management Architecture

Figure 1 illustrates the network incident management architecture. Two key components for the incident management are incident management client and incident management server.

Incident management server can be deployed in network analytics platform, controllers and provides functionalities such as incident identification, report, diagnosis, resolution, querying for incident lifecycle management.

Incident management client can be deployed in the network OSS or other business systems of operators and invokes the functionalities provided by incident management server to meet the business requirements of fault management.

A typical workflow of network incident management is as follows:

* Some alarms or abnormal operations, network performance metrics are reported from the network. Incident management server receives these alarms/abnormal operations/metrics and try to analyze the correlation of them, if the incidents are identified, it will be reported to the client. The impact of network services will be also analyzed and will update the incident.

* Incident management client receives the incident raised by server, and acknowledge it. Client may invoke the "incident diagnose" rpc to diagnose this incident to find the root causes.

* If the root causes have been found, the client can resolve this incident by invoking the 'incident resolve' rpc operation, dispatching a ticket or using other functions (e.g. routing calculation,configuration)

4.1. Interworking with Alarm Management

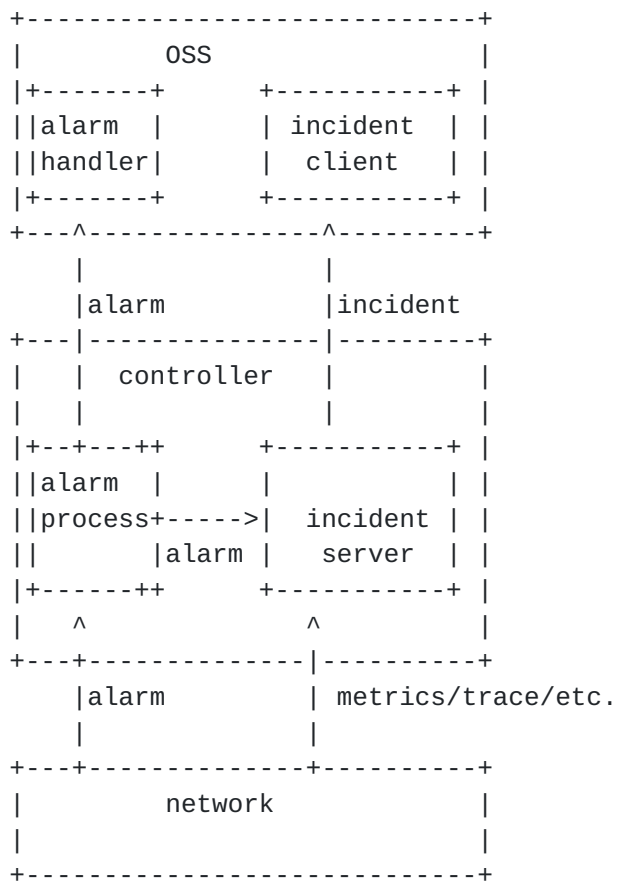


Figure 2: Interworking with alarm management

YANG model for the alarm management [RFC8632] defines a standard interface to manage the lifecycle of alarms. Alarms represent the undesirable state of network resources, alarm data model also defines the root causes and impacted services fields, but there may lack sufficient information to determine them in lower layer system (mainly in devices level), so alarms do not always tell the status of services or the root causes. As described in [RFC8632], alarm management act as a starting point for high-level fault management. While incident management often works at the network level, so it's possible to have enough information to perform correlation and service impact analysis. Alarms can work as one of data sources of incident management and may be aggregated into few amount of incidents by correlation analysis, network service impact and root causes may be determined during incident process.

Incident also contains some related alarms,if needed users can query the information of alarms by alarm management interface [RFC8632]. In some cases, e.g. cutover scenario, incident server may use alarm management interface [RFC8632] to shelve some alarms.

Alarm management may keep the original process, alarms are reported from network to network controller or analytics and then reported to upper layer system(e.g. OSS). Upper layer system may store these

alarms and provide the information for fault analysis (e.g. deeper analysis based on incident).

Compared with alarm management, incident management provides not only incident reporting but also diagnosis and resolution functions, it's possible to support self-healing and may be helpful for single-domain closed-loop control.

Incident management is not a substitute for alarm management. Instead, they can work together to implement fault management.

4.2. Interworking with SAIN

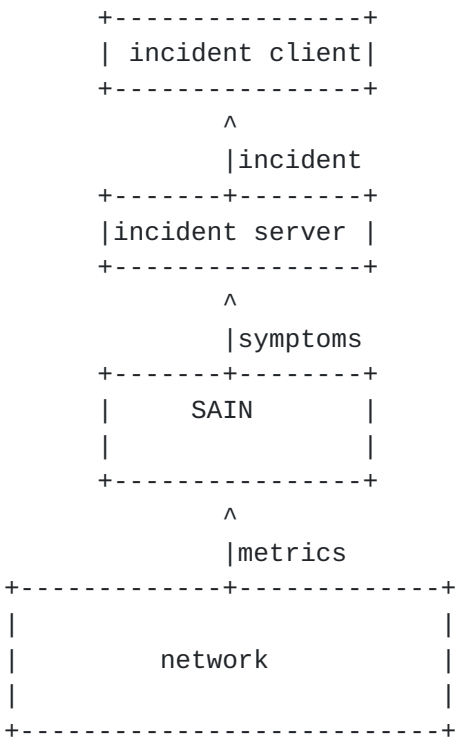


Figure 3: Interworking with SAIN

SAIN [RFC9417] defines the architecture of network service assurance. A network service can be decomposed into some sub-services, and some metrics can be monitored for sub-services. For example, a tunnel service can be decomposed into some peer tunnel interface sub-services and IP connectivity sub-service. If some metrics are evaluated to indicate unhealthy for specific sub-service, some symptoms will be present. Incident server may identify the incident based on symptoms, and then report it to upper layer system. So, SAIN can be one way to identify incident, services, sub-services and metrics can be preconfigured via APIs defined by service assurance YANG model [RFC9418] and incident will be reported if symptoms match the condition of incident.

4.3. Relationship with RFC8969

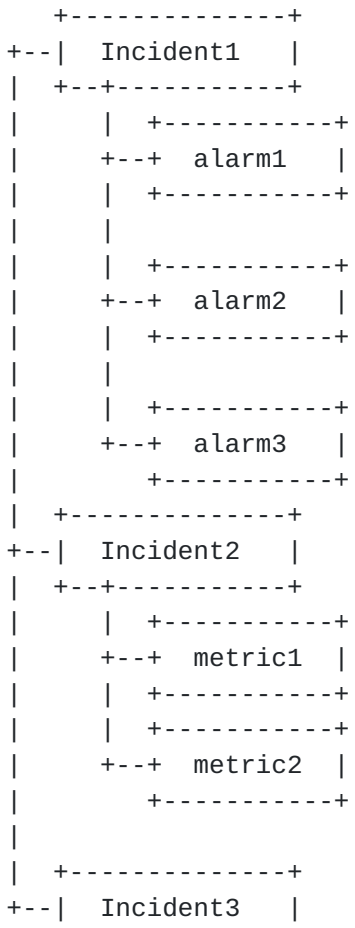
[RFC8969] defines a framework for network automation using YANG, this framework breaks down YANG modules into three layers, service layer, network layer and device layer, and contains service deployment, service optimization/assurance, and service diagnosis. Incident works at the network layer and aggregates alarms, metrics and other information from device layer, it's helpful to provfide service assurance. And the incident diagnosis may be one way of service diagnosis.

4.4. Relationship with Trace Context

W3C defines a common trace context [W3C-Trace-Context] for distributed system tracing, [I-D.rogaglia-netconf-trace-ctx-extension] defines a netconf extension for [W3C-Trace-Context] and [I-D.quilbeuf-opsawg-configuration-tracing] defines a mechanism for configuration tracing. If some errors occur when services are deploying, it's very easy to identify these errors by distributed system tracing, and an incident should be reported.

5. Functional Interface Requirements between the Client and the Server

5.1. Incident Identification



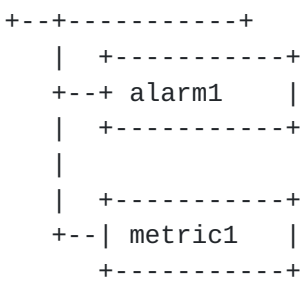


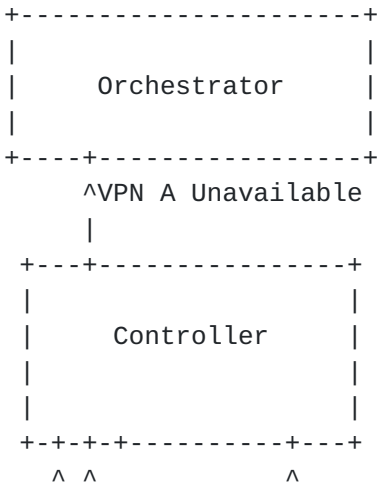
Figure 4: Incident Identification

As described in Figure 4, multiple alarms, metrics, or hybrid can be aggregated into an incident after analysis.

The network incident management server MUST be capable of identifying incidents. Multiple alarms, metrics and other information are reported to incident server, and the server must analyze it and find out the correlations of them, if the correlation match the incident rules, incident will be identified and reported to the client. Service impact analysis will be performed if an indent is identified, and the content of incident will be updated if impacted network services are detected.

AI/ML may be used to identify the incident. Expert system and online learning can help AI to identify the correlation of alarms, metrics and other information by time-base correlation algorithm, topo-based correlation algorithm, etc. For example, if interface is down, then many protocol alarms will be reported, AI will think these alarms have some correlations. These correlations will be put into knowledge base, and the incident will be identified faster according to knowledge base next time.

As mentioned above, SAIN is another way to implement incident identification. Observation timestamp defined in [I-D.tgraf-yang-push-observation-time] and trace context defined in [W3C-Trace-Context] may be helpful for incident identification.



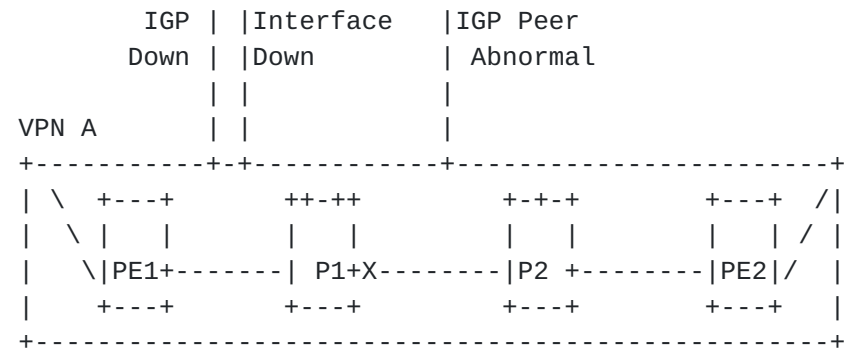


Figure 5: Example 1 of Incident Identification

As described in Figure 5, vpn a is deployed from PE1 to PE2, if a interface of P1 is going down, many alarms are triggered, such as interface down, igp down, and igp peer abnormal from P2.

These alarms are aggregated and analyzed by the controller/incident management server, and then the incident 'vpn unavailable' is triggered by the controller/incident management server.

Note that incident management server can rely on data correlation technology such as service impact analysis and data analytic component to evaluate the real effect on the relevant service and understand whether lower level or device level network anomaly, e.g., igp down, has impact on the service.

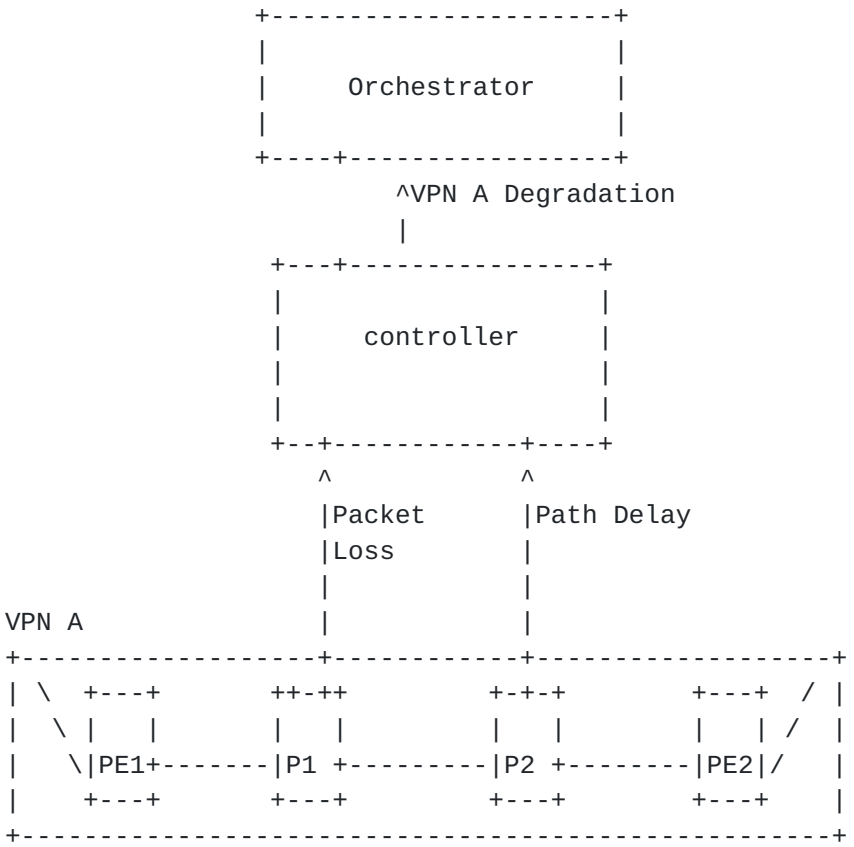


Figure 6: Example 2 of Incident Identification

As described in Figure 6, controller collect the network metrics from network elements, it finds the packet loss of P1 and the path delay of P2 exceed the thresholds, an incident 'VPN A degradation' may be triggered after service impact analysis.

5.2. Incident Diagnosis

After an incident is reported to the network incident management client, the client MAY diagnose the incident to determine the root cause. Some diagnosis operations may affect the running network services. The client can choose not to perform that diagnosis operation after determining the impact is not trivial. The network incident management server can also perform self-diagnosis. However, the self-diagnosis MUST not affect the running network services. Possible diagnosis methods include link reachability detection, link quality detection, alarm/log analysis, and short-term fine-grained monitoring of network quality metrics, etc.

5.3. Incident Resolution

After the root cause is diagnosed, the client MAY resolve the incident. The client MAY choose resolve the incident by invoking other functions, such as routing calculation function, configuration function, dispatching a ticket or asking the server to resolve it. Generally, the client would attempt to directly resolve the root cause. If the root cause cannot be resolved, an alternative solution SHOULD be required. For example, if an incident caused by a physical component failure, it cannot be automatically resolved, the standby link can be used to bypass the faulty component.

Incident server will monitor the status of incident, if the faults are fixed, the server will update the status of incident to 'cleared', and report the updated incident to the client.

Incident resolution may affect the running network services. The client can choose not to perform those operations after determining the impact is not trivial.

6. Incident Data Model Concepts

6.1. Identifying the Incident Instance

An incident ID is used as an identifier of an incident instance, if an incident instance is identified, a new incident ID is created. The incident ID MUST be unique in the whole system.

6.2. The Incident Lifecycle

6.2.1. Incident Instance Lifecycle

From an incident instance perspective, an incident can have the following lifecycle: 'raised', 'updated', 'cleared'. When an incident is generated, the status is 'raised'. If the status changes after the incident is generated, (for example, self-diagnosis, diagnosis command issued by the client, or any other condition causes the status to change but does not reach the 'cleared' level.) , the status changes to 'updated'. When an incident is successfully resolved, the status changes to 'cleared'.

6.2.2. Operator Incident Lifecycle

From an operator perspective, the lifecycle of an incident instance includes 'acknowledged', 'diagnosed', and 'resolved'. When an incident instance is generated, the operator SHOULD acknowledge the incident. And then the operator attempts to diagnose the incident (for example, find out the root cause and affected components). Diagnosis is not mandatory. If the root cause and affected components are known when the incident is generated, diagnosis is not required. After locating the root cause and affected components, operator can try to resolve the incident.

7. Incident Data Model Design

7.1. Overview

There are two YANG modules in the model. The first module, "ietf-incident-type", provides common definitions such as incident domain, incident category, incident priority. The second module, "ietf-incident", defines technology independent abstraction of network incident construct for alarm, log, performance metrics, etc. The information reported in the incident include Root cause, priority,impact, suggestion, etc. At the top of "ietf-incident" module is the Network Incident. Network incident is represented as a list and indexed by "incident-id". Each Network Incident is associated with a service instance, domain and sources. Under sources, there is one or more sources. Each source corresponds to node defined in the network topology model and network resource in the network device,e.g., interface. In addition, "ietf-incident" support one general notification to report incident state changes and three rpcs to manage the network incidents.

```
module: ietf-incident
+--ro incidents
   +--ro incident* [incident-id]
          +--ro incident-id string
          +--ro csn? uint64
          +--ro service-instance* string
          +--ro name? string
          +--ro type? enumeration
          +--ro domain? identityref
```

```
          +--ro priority? int:incident-priority
          +--ro status? enumeration
          +--ro ack-status? enumeration
          +--ro category? identityref
          +--ro detail? string
          +--ro resolve-advice? string
          +--ro sources
          ...
          +--ro root-causes
          ...
          +--ro root-events
          ...
          +--ro events
          ...
          +--ro raise-time? yang:date-and-time
          +--ro occur-time? yang:date-and-time
          +--ro clear-time? yang:date-and-time
          +--ro ack-time? yang:date-and-time
          +--ro last-updated? yang:date-and-time
rpcs:
  +---x incident-acknowledge
  ...
  +---x incident-diagnose
  ...
  +---x incident-resolve

notifications:
  +---n incident-notification
         +--ro incident-id?
                        -> /inc:incidents/inc:incident/inc:incident-id
         ...
         +--ro time? yang:date-and-time
```

7.2. Incident Notifications

```
notifications:
  +---n incident-notification
         +--ro incident-id?
                        -> /inc:incidents/inc:incident/inc:incident-id
         +--ro csn? uint64
         +--ro service-instance* string
         +--ro name? string
         +--ro type? enumeration
         +--ro domain? identityref
         +--ro priority? int:incident-priority
         +--ro status? enumeration
         +--ro ack-status? enumeration
         +--ro category? identityref
         +--ro detail? string
         +--ro resolve-advice? string
         +--ro sources
```

```
|  +--ro source* [node-ref]
|     +--ro node-ref  leafref
|     +--ro network-ref?  -> /nw:networks/network/network-id
|     +--ro resource* [name]
|        +--ro name al:resource
+--ro root-causes
|  +--ro root-cause* [node-ref]
|     +--ro node-ref  leafref
|     +--ro network-ref?  -> /nw:networks/network/network-id
|     +--ro resource* [name]
|     |  +--ro name al:resource
|     |  +--ro cause-name? string
|     |  +--ro detail? string
|     +--ro cause-name? string
|     +--ro detail? string
+--ro root-events
|  +--ro root-event* [type event-id]
|     +--ro type -> ../../../events/event/type
|     +--ro event-id leafref
+--ro events
|  +--ro event* [type event-id]
|     +--ro type enumeration
|     +--ro event-id string
|     +--ro (event-type-info)?
|        +--:(alarm)
|        |  +--ro alarm
|        |     +--ro resource? leafref
|        |     +--ro alarm-type-id? leafref
|        |     +--ro alarm-type-qualifier? leafref
|        +--:(notification)
|        +--:(log)
|        +--:(KPI)
|        +--:(unknown)
+--ro time? yang:date-and-time
```

A general notification, incident-notification, is provided here. When an incident instance is identified, the notification will be sent. After a notification is generated, if the network incident management server performs self diagnosis or the client uses the interfaces provided by the network incident management server to deliver diagnosis and resolution actions, the notification update behavior is triggered, for example, the root cause objects and affected objects are updated. When an incident is successfully resolved, the status of the incident would be set to 'cleared'.

7.3. Incident Acknowledge

```
+---x incident-acknowledge
|  +---w input
|  |  +---w incident-id*
|  |          -> /inc:incidents/inc:incident/inc:incident-id
```

After an incident is generated, updated, or cleared, (In some scenarios where automatic diagnosis and resolution are supported, the status of an incident may be updated multiple times or even automatically resolved.) The operator needs to confirm the incident to ensure that the client knows the incident.

The incident-acknowledge rpc can confirm multiple incidents at a time

7.4. Incident Diagnose

```
+---x incident-diagnose
|  +---w input
|  |  +---w incident-id*
|  |          -> /inc:incidents/inc:incident/inc:incident-id
```

After an incident is generated, incident diagnose rpc can be used to diagnose the incident and locate the root causes. Diagnosis can be performed on some detection tasks, such as BFD detection, flow detection, telemetry collection, short-term threshold alarm, configuration error check, or test packet injection.

After the diagnosis is performed, a incident update notification will be triggered to report the latest status of the incident.

7.5. Incident Resolution

```
+---x incident-resolve
 +---w input
 |  +---w incident-id*
 |          -> /inc:incidents/inc:incident/inc:incident-id
```

After the root causes and impacts are determined, incident-resolve rpc can be used to resolve the incident (if the server can resolve it). How to resolve an incident instance is out of the scope of this document.

Incident resolve rpc allows multiple incident instances to be resolved at a time. If an incident instance is successfully resolved, a notification will be triggered to update the incident status to 'cleared'. If the incident content is changed during this process, a notification update will be triggered.

8. Network Incident Management YANG Module

This module imports types defined in [RFC6991], [RFC8345], [RFC8632].

```
<CODE BEGINS> file "ietf-incident@2024-03-02.yang"
module ietf-incident {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-incident";
  prefix inc;
```

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

 import ietf-alarms {
   prefix al;
   reference
     "RFC 8632: A YANG Data Model for Alarm Management";
 }

 import ietf-network {
 prefix nw;
 reference
       "RFC 8345: A YANG Data Model for Network Topologies";
 }

 organization
   "IETF OPSAWG Working Group";
 contact
   "WG Web:   <https://datatracker.ietf.org/wg/opsawg/>;
    WG List:  <mailto:opsawg@ietf.org>
    Author:   Chong Feng  <mailto:frank.fengchong@huawei.com>
    Author:   Tong Hu  <mailto:hutong@cmhi.chinamobile.com>
    Author:   Luis Miguel Contreras Murillo <mailto:
              luismiguel.contrerasmurillo@telefonica.com>
    Author :  Qin Wu   <mailto:bill.wu@huawei.com>
    Author:   Chaode Yu   <mailto:yuchaode@huawei.com>
    Author:   Nigel Davis   <mailto:ndavis@ciena.com>";

description
   "This module defines the interfaces for incident management
    lifecycle.

    This module is intended for the following use cases:
    * incident lifecycle management:
      - incident report: report incident instance to client
                         when an incident instance is detected.
      - incident acknowledge: acknowledge an incident instance.
      - incident diagnose: diagnose an incident instance.
      - incident resolve: resolve an incident instance.

    Copyright (c) 2024 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).

   This version of this YANG module is part of RFC XXXX
   (https://www.rfc-editor.org/info/rfcXXXX); ; see the RFC
   itself for full legal notices.

   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 (RFC 2119) (RFC 8174) when, and only when,
   they appear in all capitals, as shown here. ";

revision 2024-03-02 {
  description "Merge incident yang with incident type yang
  and fix broken ref.";
  reference "RFC XXX: YANG module for network incident management.";
}

  //identities
identity incident-domain {
  description "The abstract identity to indicate the domain of
               an incident.";
}
identity single-domain {
  base incident-domain;
  description "single domain.";
}
identity access {
  base single-domain;
  description "access domain.";
}
identity ran {
  base access;
  description "radio access network domain.";
}
identity transport {
  base single-domain;
  description "transport domain.";
}
identity otn {
  base transport;
  description "optical transport network domain.";
}
identity ip {
  base single-domain;
  description "ip domain.";
}
identity ptn {
  base ip;
  description "packet transport network domain.";
}
```

```
identity cross-domain {
  base incident-domain;
  description "cross domain.";
}
identity incident-category {
  description "The abstract identity for incident category.";
}
identity device {
  base incident-category;
  description "device category.";
}
identity power-environment {
  base device;
  description "power environment category.";
}
identity device-hardware {
  base device;
  description "hardware of device category.";
}
identity device-software {
  base device;
  description "software of device category";
}
identity line {
  base device-hardware;
  description "line card category.";
}
identity maintenance {
  base incident-category;
  description "maintenance category.";
}
identity network {
  base incident-category;
  description "network category.";
}
identity protocol {
  base incident-category;
  description "protocol category.";
}
identity overlay {
  base incident-category;
  description "overlay category";
}
identity vm {
  base incident-category;
  description "vm category.";
}
identity event-type {
      description "The abstract identity for Event type";
}
```

```
identity alarm {
      base event-type;
      description "alarm event type.";
}
identity notif {
      base event-type;
      description "Notification event type.";
}
identity log {
      base event-type;
      description "Log event type.";
}
identity KPI {
      base event-type;
      description "KPI event type.";
}
identity unknown {
      base event-type;
      description "Unknown event type.";
}
identity incident-class {
      description "The abstract identity for Incident category.";
}
identity problem {
      base incident-class;
      description
        "It indicates the class of the incident is a problem
        (i.e.,cause of the incident) for example an interface
        fails to work.";
}
identity sla-violation {
      base incident-class;
      description
        "It indicates the class of the incident is a sla
        violation, for example high CPU rate may cause
        a fault in the future.";
}
//typedefs
typedef incident-priority {
  type enumeration {
    enum critical {
      description "the incident MUST be handled immediately.";
    }
    enum high {
      description "the incident should be handled as soon as
                   possible.";
    }
    enum medium {
      description "network services are not affected, or the
                   services are slightly affected,but corrective
                   measures need to be taken.";
```

```
    }
    enum low {
      description "potential or imminent service-affecting
                   incidents are detected,but services are
                   not affected currently.";
    }
  }
  description "define the priority of incident.";
}

typedef incident-ref {
      type leafref {
        path "/inc:incidents/inc:incident/inc:incident-id";
      }
  description "reference a network incident.";
}

//groupings
grouping root-cause-info {
      description "The information of root cause.";
      leaf cause-name {
    type string;
        description "the name of cause";
      }
      leaf detail {
    type string;
        description "the detail information of the cause.";
      }
}
grouping resources-info {
  description "the grouping which defines the network
               resources of a node.";
  uses nw:node-ref;
  list resource {
    key name;
    description "the resources of a network node.";
    leaf name {
       type al:resource;
       description "network resource name.";
    }
  }
}

grouping incident-time-info {
  description "the grouping defines incident time information.";
  leaf raise-time {
    type yang:date-and-time;
    description "the time when an incident instance is raised.";
  }
  leaf occur-time {
    type yang:date-and-time;
```

```
    description "the time when an incident instance occurs.
                 It's the occur time of the first event during
                 incident detection.";
  }
  leaf clear-time {
    type yang:date-and-time;
    description "the time when an incident instance is
                 resolved.";
  }
  leaf ack-time {
    type yang:date-and-time;
    description "the time when an incident instance is
                 acknowledged.";
  }
  leaf last-updated {
    type yang:date-and-time;
    description "the latest time when an incident instance is
                 updated";
  }
}

grouping incident-info {
  description "the grouping defines the information of an
               incident.";
  leaf incident-no {
    type uint64;
    mandatory true;
    description "The unique identifier of the incident instance.";
  }
  leaf-list service-instance {
    type string;
    description "the related network service instances of
                 the incident instance.";
  }
  leaf name {
    type string;
    mandatory true;
    description "the name of an incident.";
  }
  leaf type {
    type identityref {
    base incident-class;
    }
  mandatory true;
    description "The type of an incident.";
  }
  leaf domain {
    type identityref {
      base incident-domain;
    }
    mandatory true;
```

```
  description "the domain of an incident.";
}
leaf priority {
  type incident-priority;
  mandatory true;
  description "the priority of an incident instance.";
}

leaf status {
  type enumeration {
    enum raised {
      description "an incident instance is raised.";
    }
    enum updated {
      description "the information of an incident instance
                   is updated.";
    }
    enum cleared {
      description "an incident is cleared.";
    }
  }
  default raised;
  description "The status of an incident instance.";
}
leaf ack-status {
  type enumeration {
    enum acknowledged {
      description "The incident has been acknowledged by user.";
    }
    enum unacknowledged {
      description "The incident hasn't been acknowledged.";
    }
  }
  default unacknowledged;
  description "the acknowledge status of an incident.";
}
leaf category {
  type identityref {
    base incident-category;
  }
  mandatory true;
  description "The category of an incident.";
}
leaf detail {
  type string;
  description "detail information of this incident.";
}
leaf resolve-advice {
  type string;
  description "The advice to resolve this incident.";
}
```

```
container sources {
  description "The source components.";
  list source {
    key node-ref;
    uses resources-info;
    min-elements 1;
    description "The source components of incident.";
  }
}

container root-causes{
  description "The root cause objects.";
  list root-cause {
    key node-ref;
    description "the root causes of incident.";
    uses resources-info {
      augment resource {
        description "augment root cause information.";
        //if root cause object is a resource of a node
        uses root-cause-info;
      }
    }
    //if root cause object is a node
    uses root-cause-info;
  }
}
container root-events {
  description "the root events of the incident.";
  list root-event {
    key "type event-id";
    description "the root event of the incident.";
    leaf type {
      type leafref {
        path "../../../events/event/type";
      }
      description "the event type.";
    }
    leaf event-id {
      type leafref {
        path "../../../events/event[type = current()/../type]"
            +"/event-id";
      }
      description "the event identifier, such as uuid,
                   sequence number, etc.";
    }
  }
}
container events {
  description "related events.";
  list event {
    key "type event-id";
```

```
description "related events.";
leaf type {
  type identityref {
          base event-type;
          }
  description "event type.";
}
leaf event-id {
  type string;
  description "the event identifier, such as uuid,
               sequence number, etc.";
}
choice event-type-info {
  description "event type information.";
  case alarm {
    when "derived-from-or-self(type, 'alarm')" {
    description
    "Only applies when type is alarm.";
    }
    container alarm {
      description "alarm type event.";
      leaf resource {
        type leafref {
          path "/al:alarms/al:alarm-list/al:alarm"
              +"/al:resource";
        }
        description "network resource.";
        reference "RFC 8632: A YANG Data Model for Alarm
                   Management";
      }
      leaf alarm-type-id {
        type leafref {
          path "/al:alarms/al:alarm-list/al:alarm"
              +"[al:resource = current()/../resource]"
              +"/al:alarm-type-id";
        }
        description "alarm type id";
        reference "RFC 8632: A YANG Data Model for Alarm
                    Management";
      }
      leaf alarm-type-qualifier {
        type leafref {
          path "/al:alarms/al:alarm-list/al:alarm"
              +"[al:resource = current()/../resource]"
              +"[al:alarm-type-id = current()/.."
              +"/alarm-type-id]/al:alarm-type-qualifier";
        }
        description "alarm type qualitifier";
        reference "RFC 8632: A YANG Data Model for Alarm
                   Management";
      }
```

```
          }
        }
        case notification {
          //TODO
        }
        case log {
        //TODO
        }
        case KPI {
        //TODO
        }
        case unknown {
        //TODO
        }
      }
    }

  }

}

//data definitions
container incidents {
  config false;
  description "the information of incidents.";
  list incident {
    key incident-no;
    description "the information of incident.";
    leaf incident-id {
      type string;
      description "the qualifier of an incident instance type.";
    }
    uses incident-info;
    uses incident-time-info;
  }
}

// notifications
notification incident-notification {
  description "incident notification. It will be triggered when
               the incident is raised, updated or cleared.";
  leaf incident-id {
        type incident-ref;
    description "the identifier of an incident instance.";
  }
  uses incident-info;
  leaf time {
    type yang:date-and-time;
    description "occur time of an incident instance.";
  }
}
```

```
  // rpcs
  rpc incident-acknowledge {
    description "This rpc can be used to acknowledge the specified
                 incidents.";
    input {
      leaf-list incident-id {
        type incident-ref;
        description "the identifier of an incident instance.";
      }
    }
  }
  rpc incident-diagnose {
    description "This rpc can be used to diagnose the specified
                 incidents. The result of diagnosis will be reported
                 by incident notification.";
    input {
      leaf-list incident-id {
        type incident-ref;
        description
          "the identifier of an incident instance.";
      }
    }
  }

  rpc incident-resolve {
    description "This rpc can be used to resolve the specified
                 incidents. The result of resolution will be reported
                 by incident notification.";
    input {
      leaf-list incident-id {
        type incident-ref;
        description
          "the identifier of an incident instance.";
      }
    }
  }
}
<CODE ENDS>
```

9. Security Considerations

The YANG modules specified in this document define a schema for data that is designed to be accessed via network management protocol such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

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). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

<<TBC>>

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. These are the subtrees and data nodes and their sensitivity/vulnerability:

<<TBC>>

Some of the RPC operations in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations. These are the operations and their sensitivity/vulnerability:

<<TBC>>

10. IANA Considerations

10.1. The "IETF XML" Registry

This document requests IANA to register one XML namespace URN in the "ns" subregistry within the "IETF XML Registry" [RFC3688]:

URI: urn:ietf:params:xml:ns:yang:ietf-incident
Registrant Contact: The IESG.
XML: N/A, the requested URIs are XML namespaces.

10.2. The "YANG Module Names" Registry

This document requests IANA to register one module name in the 'YANG Module Names' registry, defined in [RFC6020].

Name: ietf-incident
Maintained by IANA? N
Namespace: urn:ietf:params:xml:ns:yang:ietf-incident
Prefix: inc
Reference: RFC XXXX
// RFC Ed.: replace XXXX and remove this comment

Acknowledgments

The authors would like to thank Mohamed Boucadair, Robert Wilton, Benoit Claise, Oscar Gonzalez de Dios, Adrian Farrel, Mahesh Jethanandani, Balazs Lengyel, Bo Wu, Qiufang Ma, Haomian Zheng, YuanYao Wei Wang, Peng Liu, Zongpeng Du, Zhengqiang Li, Andrew Liu , Joe Clark, Roland Scott, Alex Huang Feng, Kai Gao, Jensen Zhang, Ziyang Xing for their valuable comments and great input to this work.

References

Normative References

[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/rfc/rfc2119>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/rfc/rfc3688>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/rfc/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/rfc/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/rfc/rfc6242>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/rfc/rfc6991>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/rfc/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/rfc/rfc8174>.

[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/rfc/rfc8341>.

[RFC8345] Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A YANG Data Model for Network Topologies", RFC 8345, DOI 10.17487/RFC8345, March 2018, <https://www.rfc-editor.org/rfc/rfc8345>.

[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/rfc/rfc8446>.

[RFC8632] Vallin, S. and M. Bjorklund, "A YANG Data Model for Alarm Management", RFC 8632, DOI 10.17487/RFC8632, September 2019, <https://www.rfc-editor.org/rfc/rfc8632>.

Informative References

[BERT] "BERT (language model)", n.d., <https://en.wikipedia.org/wiki/BERT_(language_model)>.

[I-D.ietf-ippm-pam] Mirsky, G., Halpern, J. M., Min, X., Clemm, A., Strassner, J., and J. François, "Precision Availability Metrics for Services Governed by Service Level Objectives (SLOs)", Work in Progress, Internet-Draft, draft-ietf-ippm-pam-09, 1 December 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-ippm-pam-09>.

[I-D.quilbeuf-opsawg-configuration-tracing] Quilbeuf, J., Claise, B., Graf, T., Lopez, D., and S. Qiong, "External Trace ID for Configuration Tracing", Work in Progress, Internet-Draft, draft-quilbeuf-opsawg-configuration-tracing-02, 10 July 2023, <https://datatracker.ietf.org/doc/html/draft-quilbeuf-opsawg-configuration-tracing-02>.

[I-D.rogaglia-netconf-trace-ctx-extension] Gagliano, R., Larsson, K., and J. Lindblad, "NETCONF Extension to support Trace Context propagation", Work in Progress, Internet-Draft, draft-rogaglia-netconf-trace-ctx-extension-03, 6 July 2023, <https://datatracker.ietf.org/doc/html/draft-rogaglia-netconf-trace-ctx-extension-03>.

[I-D.tgraf-yang-push-observation-time] Graf, T., Claise, B., and A. H. Feng, "Support of Network Observation Timestamping in YANG Notifications", Work in Progress, Internet-Draft, draft-tgraf-yang-push-observation-time-00, 4 March 2023, <https://datatracker.ietf.org/doc/html/draft-tgraf-yang-push-observation-time-00>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language",

RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/rfc/rfc7950>.

[RFC8969] Wu, Q., Ed., Boucadair, M., Ed., Lopez, D., Xie, C., and L. Geng, "A Framework for Automating Service and Network Management with YANG", RFC 8969, DOI 10.17487/RFC8969, January 2021, <https://www.rfc-editor.org/rfc/rfc8969>.

[RFC9375] Wu, B., Ed., Wu, Q., Ed., Boucadair, M., Ed., Gonzalez de Dios, O., and B. Wen, "A YANG Data Model for Network and VPN Service Performance Monitoring", RFC 9375, DOI 10.17487/RFC9375, April 2023, <https://www.rfc-editor.org/rfc/rfc9375>.

[RFC9408] Boucadair, M., Ed., Gonzalez de Dios, O., Barguil, S., Wu, Q., and V. Lopez, "A YANG Network Data Model for Service Attachment Points (SAPs)", RFC 9408, DOI 10.17487/RFC9408, June 2023, <https://www.rfc-editor.org/rfc/rfc9408>.

[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/rfc/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/rfc/rfc9418>.

[TMF724A] "Incident Management API Profile v1.0.0", 2023, <https://www.tmforum.org/resources/standard/tmf724a-incident-management-api-profile-v1-0-0/>.

[W3C-Trace-Context] "W3C Recommendation on Trace Context", 2021, <https://www.w3.org/TR/2021/REC-trace-context-1-20211123/>.

Appendix A. Changes between Revisions

v02 - v03 * Fix pyang compilation issue and yang lint issue.

v01 - v02 * Fix Broken ref by using node-ref defined in RFC8345.

* Update YANG data model based on issues raised in issue tracker of the github.

* Shorten the list of authors to 5 based on chairs' comment and move additional authors to top 3 contributors.

v00 - v01

* Merge ietf-incident-type.yang into ietf-incident.yang

* Fix enumeration on leaf type

* Clarify the scope in the abstract and introduction and make the scope focus on YANG data model

* Provide text around figure 5 to clarify how the incident server know the real effect on the relevant services.

* Other editorial changes.

v00 (draft-feng-nmop-network-incident-yang)

* Change draft name from draft-feng-opsawg-incident-management into draft-feng-nmop-netwrok-incident-yang

* Change title into A YANG Data Model for Network Incident Management

* open issues is tracked in https://github.com/billwuqin/network-incident/issues

v03 - v04 (draft-feng-opsawg-incident-management)

* Update incident defintion based on TMF incident API profile specification.

* Update use case on Multi-layer Fault Demarcation based on side meeting discussion and IETF 119 session discussion.

* Update section 5.1 to explain how network incident is generated based on other factors.

* Add one new use cases on Security Events noise reduction based on Situation Awareness.

* Other Editorial changes.

v02 - v03 (draft-feng-opsawg-incident-management)

* Add one new use cases on Incident Generation.

* Add reference to Precision Availability Metric defined in IPPM PAM WG document.

v01 - v02

* A few Editorial change to YANG data models in section 8.

* Add some text to the model design overview.

* Revise sample use cases section to focus on two key use cases.

* Motivation and goal clarification in the introduction section.

v00 - v01 (draft-feng-opsawg-incident-management)

* Modify the introduction.

* Rename incident agent to incident server.

* Add the interworking with alarm management.

* Add the interworking with SAIN.

* Add the relationship with RFC8969.

* Add the relationship with observation timestamp and trace context.

* Clarify the incident identification process.

* Modify the work flow of incident diagnosis and resolution.

* Remove identities and typedefs from ietf-incident YANG module, and create a new YANG module called ietf-incident-types.

* Modify ietf-incident YANG module, for example, modify incident-diagnose rpc and incident-resolve rpc.

Contributors

Thomas Graf
Swisscom
Binzring 17CH-8045
CH- Zurich
Switzerland
Email: thomas.graf@swisscom.com

Zhenqiang Li
CMCC
Email: li_zhenqiang@hotmail.com

Yanlei Zheng
China Unicom
Email: zhengyanlei@chinaunicom.cn

Yunbin Xu
CAICT
Email: xuyunbin@caict.ac.cn

Xing Zhao

CAICT
Email: zhaoxing@caict.ac.cn

Chaode Yu
Huawei
Email: yuchaode@huawei.com

MingShuang Jin
Huawei Technologies
Email: jinmingshuang@huawei.com

Chunchi Liu
Huawei Technologies
Email: liuchunchi@huawei.com

Aihua Guo
Futurewei Technologies
Email: aihuaguo.ietf@gmail.com

Zhidong Yin
Huawei
Email: yinzhidong@huawei.com

Guoxiang Liu
Huawei
Email: liuguoxiang@huawei.com

Kaichun Wu
Huawei
Email: wukaichun@huawei.com

Authors' Addresses

Tong Hu
CMCC
Building A01, 1600 Yuhangtang Road, Wuchang Street, Yuhang District
Hangzhou
311121
China
Email: hutong@cmhi.chinamobile.com

Luis Miguel Contreras Murillo
Telefonica I+D

Madrid
Spain
Email: luismiguel.contrerasmurillo@telefonica.com

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing
210012
China
Email: bill.wu@huawei.com

Nigel Davis
Ciena
Email: ndavis@ciena.com

Chong Feng
Email: fengchonglly@gmail.com