CCAMP Working Group
Internet-Draft
Intended status: Standards Track
Expires: 25 November 2026

B. Y. Yoon
ETRI
Y. You
woori-net

24 May 2026

A YANG Data Model of Performance Management Streaming
draft-yoon-ccamp-pm-streaming-05

Abstract

This document specifies a YANG data model for the Performance Management (PM) Collection function requirements defined in ITU-T G.7710, which processes raw performance data measured at a network node and delivers the resulting data to clients using the IETF push-based streaming model.

About This Document

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

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-yoon-ccamp-pm-streaming/.

Source for this draft and an issue tracker can be found at https://github.com/https://github.com/binyeongyoon-ietf/ietf-pm-streaming.

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 25 November 2026.

Table of Contents

1. Introduction

Performance Management (PM) data generated by a Network Element (NE) undergoes a systematic processing pipeline that transforms raw observations into actionable operational insights. Traditionally, PM data has been managed through pull-based mechanisms such as SNMP polling, but the increasing scale and dynamic nature of modern networks require a more efficient, streaming-oriented approach.

This document distinguishes three functional stages of the PM processing pipeline -- Measurement, Collection, and Reporting -- and defines YANG data models for the Collection stage. The continuous, push-based delivery of Collection-stage data and notifications to external clients is referred to in this document as PM streaming.

The Collection-stage data and notifications defined here are designed to be carried to clients through the existing IETF subscribed-notifications framework [RFC8639], its YANG Push extension for datastore updates [RFC8641], and the NETCONF binding for dynamic subscriptions [RFC8640]. These three specifications together are referred to in the remainder of this document as the push-based subscription mechanisms.

1.1. PM Processing Pipeline in a Network Element

The PM processing pipeline considered in this document comprises the following three stages.

Stage 1 -- Measurement. This stage observes signals or protocol behaviours to produce raw measurement values. It encompasses direct observation of physical-layer signals as well as various OAM-based techniques. As examples of active measurement protocols that may be used in this stage, OWAMP [RFC4656], TWAMP [RFC5357], and STAMP [RFC8762] inject synthetic probe packets to measure one-way or round-trip delay, packet loss, and delay variation, while In-situ OAM (IOAM) [RFC9197] records performance data directly within user packets as they traverse the network.

Stage 2 -- Collection. This stage processes raw measurement data into summarised statistics over defined intervals and manages their retention. Historically, PM collection mechanisms based on SNMP MIBs used the performance-history textual conventions defined in [RFC3593], with the high-capacity variants in [RFC3705]. The Collection stage modelled in this document follows the common equipment management function requirements of ITU-T G.7710 and supports three collection types -- Counts, Snapshot, and Tidemarks -- together with threshold evaluation and the associated periodic and non-periodic events.

Stage 3 -- Reporting. This stage delivers Collection-stage data and notifications to external management systems or controllers. While traditional pull-based retrieval remains available, this document assumes the push-based subscription mechanisms introduced above ([RFC8639], [RFC8641], [RFC8640]). The role of the Collection-stage models defined here is to provide the data structures that those mechanisms carry. Typical clients of this Reporting stage include operations systems (OS), network digital twins (NDT), and AI-driven applications (AI APP), as illustrated in Figure 1.

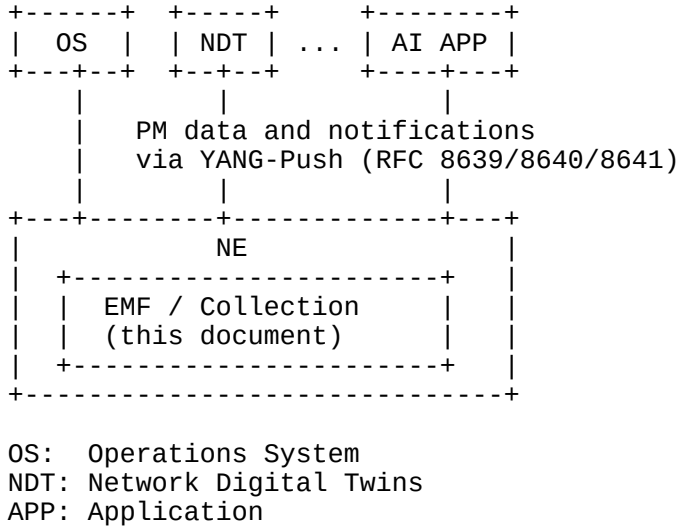

Figure 1: Streaming Network Architecture

The YANG data model specified in this document is implemented within the Equipment Management Function (EMF) of an NE, as defined in ITU-T G.7710 Section 6.2. It is exposed at the NE's management interface and is consumed by remote clients such as operations systems (OS), Physical Network Controllers (PNCs), network digital twins (NDT), and AI-driven applications. Throughout this document the term "client" refers to any such consumer of the PM streaming interface, regardless of whether it acts as a controller, an analytics engine, or an operator-facing management system.

1.2. Motivation and Scope

The legacy SNMP-based collection conventions [RFC3593] [RFC3705] and the associated pull-based polling architecture limit real-time visibility and flexible interval design at the scale of modern networks. The push-based subscription mechanisms [RFC8639] [RFC8640] [RFC8641] provide an efficient and scalable alternative for streaming management data, but those specifications define only the subscription and notification mechanisms themselves and do not prescribe the structure of the PM data being carried. A standardised YANG data model is therefore needed at the Collection stage to bridge low-level measurement data and high-level streaming subscriptions, since existing YANG models tend to focus on static configuration or simple state data and do not offer the structures required to express counts, snapshot, and tidemarks collection together with profile-based parameter grouping and threshold semantics aligned with ITU-T G.7710.

The scope of this document is therefore limited to the Collection stage. Within that scope, this document specifies YANG data models that enable: (a) flexible processing of performance metrics beyond simple counts, including Snapshot and Tidemarks, as defined in ITU-T G.7710; (b) configurable sampling and measurement intervals that are not restricted to the legacy 15-minute or 24-hour windows, together with a capability-discovery mechanism in ietf-pm-interval-capabilities that lets clients learn which intervals a given server supports; and (c) data and notification structure designed to be carried by the push-based subscription mechanisms identified above for real-time, proactive delivery of processed PM data to management systems.

The data model defined in this document realizes the Collection-stage functions described in [G7710], in particular performance-monitoring data, collection types, and threshold reporting.

The Collection-stage data and notifications defined here are intended to be carried by the existing YANG-Push framework ([RFC8639], [RFC8640], [RFC8641]); this document does not define any new subscription or notification.

2. PM parameters

2.1. Types

Performance monitoring (PM) in networks encompasses a wide variety of parameters that reflect operational health, service quality, reliability, and environmental conditions. These parameters are used across many technologies, network layers, and functional domains to enable fault management, SLA compliance, trend analysis, predictive maintenance, and operational optimization.

PM parameter types include but are not limited to:

* Classical transport and packet layer metrics: such as errored seconds (ES), severely errored seconds (SES), unavailable seconds (UAS), background block errors (BBE), background block counts (BBC), delay, jitter, and packet loss, as defined in standards like ITU-T G.7710, and others.

* Layer-specific metrics:

  - Physical layer: optical power levels, laser bias current, loss of signal

  - Data link layer: Ethernet frame errors, FCS errors

- Network layer: dropped packets, route flaps

- Transport/Service layers: MPLS LSP statistics, OTN TCM/BIP counters

* Network environment parameters: including temperature, humidity, fan speed, voltage, and airflow. These are essential for equipment safety, energy management, and predictive failure analysis.

* Energy and sustainability metrics: such as power consumption, energy efficiency indicators, and cooling utilization, aligned with emerging sustainability standards and operational efficiency goals.

* Security and integrity parameters: such as pointer justification events (PJE), synchronization loss, or intrusion anomaly flags.

* Application-aware or SLA metrics: such as service availability, throughput consistency, and application-layer latency.

* Mobile network-specific metrics: including radio link failures, handover success/failure rates, RRC connection setup time, PDCP discard rate, and throughput per bearer. These metrics are critical for monitoring the performance of RAN, core, and edge network components in 4G/5G mobile environments.

These parameters may be grouped flexibly within the YANG model using parameter profiles that reflect shared characteristics, purpose, or applicable network domains. The architecture supports extension through identity-based typing to accommodate future parameter definitions introduced by standard bodies like ITU-T, IEEE, IETF, MEF, and TM Forum.

2.2. Profiles

```
+--rw parameter-profile* [name]
   +--rw name              profile-names
   +--rw pm-parameter* [name]
      +--rw name           string
```

Figure 2: Parameter Profile Subtree

The YANG model defines the concept of a parameter profile to logically group performance parameters that are commonly measured together for a specific operational purpose. Each parameter profile is represented as a list entry keyed by a name of type profile-names (a string conforming to the format defined in the model). These

profiles serve as named collections of performance parameters and are intended to facilitate streamlined configuration, management, and reporting of measurement data across network elements and management systems.

The use of parameter profiles improves operational efficiency by allowing operators, applications, and controllers to activate or reference a coherent set of parameters using a single profile identifier. For example, the itu-transport-maintenance-15min profile may include parameters such as errored seconds (ES), severely errored seconds (SES), and unavailable seconds (UAS), which are typically monitored together for network maintenance and fault detection purposes. Similarly, the example-ip-qos-24hr profile may include delay, jitter, and loss parameters used in service quality reporting. Parameter profiles support role-based access control, operational alignment, and measurement policy abstraction, enabling network operators and analytics systems to tailor data collection and reporting according to the needs of different users and services. The profile abstraction also aligns with ITU-T G.7710, which identifies multiple classes of performance monitoring (e.g., maintenance, service-level, and compliance monitoring), each requiring specific sets of performance parameters.

By modeling profiles as list entries keyed by a structured name (profile-names type), the YANG design ensures extensibility and vendor interoperability, allowing future profiles to be defined without changes to the core data structures. This approach promotes consistent configuration and integration across multi-vendor environments and supports dynamic service assurance use cases where parameter sets may vary by service type, SLA, or operational context.

2.2.1. Naming

Parameter profiles are named to reflect their operational purpose, origin, applicable network domain, and, optionally, the primary measurement interval. This naming structure supports clarity, modularity, and automation across diverse network and service layers.

The naming follows this format:

<source>-<network>-<purpose>[-<characteristic>]

Where:

* <source>: Standards body or organization

  - Examples: itu, ieee, ietf, example, vendorX

* <network>: Network domain or layer
  - Examples: transport, access, core, ip, mpls, ethernet, otn, wdm, flexo
* <purpose>: Intended use or function
  - Examples: maintenance, qos, availability, sla, compliance, analytics
* <characteristic> (optional): Optional qualifying information
  - Examples: 15min, 24hr, high-priority

Examples:

* itu-transport-maintenance-15min
* itu-transport-qos-24hr
* ieee-access-availability
* example-ip-qos-24hr
* vendorx-otn-sla

Note: profile names that begin with ietf- are reserved for profiles that IETF actually defines; placeholder examples in this document use example- to avoid implying IETF endorsement of profiles not yet specified.

The term ’transport’ means that the profile applies to multiple technologies (e.g., OTN, MPLS-TP, Transport Ethernet, etc.).

The 15-minute interval provides granular, real-time monitoring, allowing network operators to quickly detect and address short-term issues such as spikes in latency or packet loss. It is particularly useful for ensuring compliance with Service-Level Agreements (SLAs) and for managing highly dynamic networks where rapid changes can occur. In contrast, the 24-hour interval is used for long-term performance monitoring and trend analysis, helping operators understand overall network health, detect slow-developing issues, and plan for future capacity needs. This longer interval offers a broader view of the network’s performance over a full day, making it ideal for strategic planning and infrastructure maintenance. Together, these intervals enable both immediate responses to network conditions and long-term network optimization.

2.3. Transport Common PM Parameters

Metric values of PM parameters are measured for maintenance and QoS monitoring over networks. Quality of Service (QoS) parameters are designed to assess the network's long-term ability to consistently deliver agreed-upon service quality to customers. They primarily verify performance against contractual obligations defined in service-level agreements (SLAs) over longer intervals (24 hours, monthly periods). By simultaneously measuring both directions of a bidirectional connection, QoS parameters provide a holistic view of the sustained quality experienced by users, ensuring stability and predictability.

Maintenance parameters focus on short-term monitoring and detailed analysis for operational reliability. Maintenance parameters, over intervals such as 15 minutes or 24 hours, facilitate swift responses to intermittent faults, bursts of errors, and subtle performance changes. Maintenance parameters typically involve unidirectional analysis, where each direction of transmission is monitored independently. This unidirectional approach helps network operators precisely pinpoint faults, troubleshoot intermittent issues, and perform preventive maintenance effectively.

Key PM parameters focused on circuit networks such as OTN are listed as follows. Additional parameters will be needed for packet networks.

* ES Errored Seconds
* SES Severely Errored Seconds
* BBE Background Block Errors
* BBC Background Block Count
* UAS Unavailable Seconds
* SEP Severely Errored Period
* PJE Pointer Justification Events

According to collection types, purposes, and time intervals, different parameters are used. The following three transport profiles provide comprehensive coverage for maintenance and QoS monitoring across different temporal resolutions.

2.3.1. itu-transport-maintenance-15min Profile

The itu-transport-maintenance-15min profile is designed for short-term operational monitoring and rapid fault detection. This profile utilizes all three collection types (counts, snapshot, and tidemarks) over 15-minute intervals to provide granular visibility into network performance. The profile includes core maintenance parameters: ES, SES, BBE, BBC, and UAS.

The 15-minute interval enables operators to quickly detect and respond to performance degradation, making it ideal for proactive maintenance and immediate troubleshooting. The combination of counts (for cumulative event tracking), snapshot (for instantaneous state capture), and tidemarks (for extreme value monitoring) provides a comprehensive view of network behavior within each measurement period. This profile is particularly valuable for network operations centers (NOCs) requiring real-time visibility into transport network health and for automated systems that need to trigger immediate responses to performance anomalies.

2.3.2. itu-transport-maintenance-24hr Profile

The itu-transport-maintenance-24hr profile extends the maintenance monitoring capabilities to longer-term analysis and trend identification. Similar to the 15-minute profile, it employs all three collection types (counts, snapshot, and tidemarks) but over 24-hour intervals. The parameter set includes ES, SES, BBE, BBC, UAS, and additionally PJE.

The 24-hour measurement interval allows for comprehensive daily performance analysis, enabling operators to identify patterns, trends, and recurring issues that may not be apparent in shorter intervals. The inclusion of PJE provides additional insight into synchronization-related performance issues that are more relevant over longer observation periods. This profile supports strategic maintenance planning, capacity management, and historical performance analysis. It complements the 15-minute profile by providing the broader context needed for long-term network optimization and preventive maintenance strategies.

2.3.3. itu-transport-qos-24hr Profile

The itu-transport-qos-24hr profile is specifically designed for Quality of Service monitoring and SLA compliance verification. Unlike the maintenance profiles, this profile focuses exclusively on the counts collection type over 24-hour intervals, emphasizing sustained performance assessment rather than detailed operational monitoring. The parameter set includes ES, SES, BBE, BBC, SEP, and

UAS.

The QoS profile’s exclusive use of counts collection type reflects its focus on cumulative performance over extended periods, which is essential for SLA compliance and service quality reporting. The 24-hour interval aligns with typical SLA measurement periods and provides the statistical basis for service quality assessments. The inclusion of SEP is particularly relevant for QoS monitoring as it represents sustained periods of degraded performance that directly impact service quality commitments.

2.3.4. Profile Relationships and Operational Integration

These three profiles work together to provide a comprehensive performance monitoring framework. The itu-transport-maintenance-15min profile serves as the primary operational tool for immediate network management, while the itu-transport-maintenance-24hr profile provides the analytical foundation for strategic planning and trend analysis. The itu-transport-qos-24hr profile ensures that service quality commitments are met and provides the data necessary for SLA reporting and customer assurance.

The hierarchical relationship between these profiles enables operators to correlate short-term operational events with long-term performance trends and service quality metrics. For example, a spike in ES detected by the 15-minute maintenance profile can be correlated with daily trends from the 24-hour maintenance profile and assessed against SLA thresholds defined in the QoS profile. This integrated approach supports both reactive troubleshooting and proactive network optimization while ensuring compliance with service quality commitments.

3. Periodic Measurement

3.1. Measurement Timing

```
+--rw sampling-interval* [id]
   +--rw id                  string
   +--rw interval-value?     uint32
   +--rw unit?               time-interval-unit
   +--rw measurement-interval* [id]
      +--rw id                string
      +--rw interval-value?   uint32
      +--rw unit?             time-interval-unit
```

Figure 3: Sampling and Measurement Intervals Subtree

Measurement timing parameters are key components of network performance management, offering standardized definitions for the time-related aspects of sampling, measuring, and reporting performance data. These parameters apply to the three main collection types for network equipment: counts, snapshot, and tidemarks. They include the sampling interval, measurement interval, and uniform time, all of which support consistent, accurate, and systematic performance monitoring and management.

Sampling interval defines the period at which network performance data is collected at consistent, predetermined time points. It ensures the continuous and timely capture of performance metrics, enabling accurate assessments of network conditions.

Measurement interval specifies the duration over which sampled performance data is aggregated or statistically processed. It helps manage large volumes of data by summarizing it into meaningful indicators for analysis, anomaly detection, and resource management.

Uniform time is a fixed, predefined point within each measurement interval at which a snapshot measurement is taken. It enables a consistent and instantaneous view of network performance across intervals, without requiring data aggregation. This approach facilitates quick diagnostics and synchronization across monitoring systems.

3.1.1. Use Cases

The hierarchical design of the ietf-pm-collection YANG module, where a performance parameter can be associated with one or more sampling intervals and each sampling interval can be associated with multiple measurement intervals and collection types, supports a wide range of operational objectives. A key benefit of this structure is that, even when the sampling interval is fixed, different measurement intervals can be used to derive distinct operational views of the same parameter. The collection type is selected to match the semantics of the monitored parameter: counts for cumulative event parameters, tidemarks for varying parameters where extremes are operationally significant, and snapshot for point-in-time operating state.

In a Network Operations Center (NOC), errored seconds (ES) can be sampled every second and processed with multiple measurement intervals using the counts collection type. A 1-minute measurement interval supports rapid fault indication, enabling fast recognition of service degradation. A 15-minute measurement interval supports routine maintenance monitoring and aligns with established operational practices described in ITU-T G.7710. A 24-hour measurement interval supports daily QoS reporting and provides a broader view of service quality over time.

For latency monitoring in a NOC, a parameter may be sampled every 500 milliseconds and processed using the tidemarks collection type. A 1-minute measurement interval helps detect short-lived delay spikes. A 30-minute measurement interval helps identify recurring burst patterns that affect path stability. A 24-hour measurement interval provides a daily worst-case view of path quality, supporting operational assessment of persistent delay behavior.

For digital twin applications, packet delay variation (PDV) may be sampled every 100 milliseconds and processed using the tidemarks collection type. A 1-minute measurement interval provides a synchronization stability envelope that helps the digital twin remain closely aligned with the physical network. A 5-minute measurement interval supports feedback-loop tuning by capturing short-term variation patterns that influence control adjustments. A 1-hour measurement interval supports model calibration by providing longer-span information about the range of delay variation observed in operation.

Environmental monitoring also benefits from the same hierarchical timing structure. For example, temperature may be sampled every 10 seconds and processed using the snapshot collection type. A 1-minute measurement interval supports a current operating state check. A 15-minute measurement interval supports periodic baseline comparison. A 24-hour measurement interval supports daily fleet-wide correlation at a common observation point, helping operators compare equipment behavior across systems and sites.

Similar timing structures can also support AI/ML pipelines, where short, medium, and long measurement intervals applied to the same sampled parameter provide feature sets for anomaly detection, trend analysis, and model training within a single analytics application.

These examples show that the same sampling interval can support multiple operational purposes when paired with different measurement intervals, and that the collection type should match the semantics of the monitored parameter. The hierarchical list structure, where parameters contain multiple sampling intervals and each sampling

interval defines one or more measurement intervals and collection types, supports operational flexibility, avoids configuration duplication, and enables fine-grained control of measurement strategies. The use cases summarized in Table 1 are consistent with the collection types defined in Section 3.2, where counts represent cumulative event occurrences, snapshot represents point-in-time values, and tidemarks represent interval extremes.

Client (param)	Samp.	Meas.	Coll.	Purpose
NOC (ES)	1s	1min	counts	Rapid fault alert
	1s	15min	counts	Maintenance
	1s	24hr	counts	Daily QoS report
NOC (latency)	500ms	1min	tidemarks	Delay spike detect
	500ms	30min	tidemarks	Burst pattern obs.
	500ms	24hr	tidemarks	Daily worst-case
Digital Twin	100ms	1min	tidemarks	Sync stability
(PDV)	100ms	5min	tidemarks	Feedback tuning
	100ms	1hr	tidemarks	Model calibration
NOC (temp)	10s	1min	snapshot	State check
	10s	15min	snapshot	Periodic baseline
	10s	24hr	snapshot	Fleet-wide daily

Table 1: Use cases of sampling, measurement, and collection

3.2. Collection Types

```
   +--rw collection-types
      +--rw counts
      |  +--rw transient-condition-config
      |  |  +--rw transient-threshold?   uint32
      |  +--rw standing-condition-config
      |  |  +--rw standing-threshold?    uint32
      |  |  +--rw reset-threshold?       uint32
      |  +--ro measurement-value?        uint32
      +--rw snapshot
      |  +--rw uniform-time-config
      |  |  +--rw interval-value?        uint32
      |  |  +--rw unit?                  time-interval-unit
      |  +--rw threshold-config
      |  |  +--rw high-threshold?        uint32
      |  |  +--rw low-threshold?         uint32
      |  +--ro measurement-value?        uint32
      +--rw tidemarks
         +--rw threshold-config
         |  +--rw high-threshold?        uint32
         |  +--rw low-threshold?         uint32
         +--ro high-measurement-value?   uint32
         +--ro low-measurement-value?    uint32
```

Figure 4: Collection Types Subtree

The collection types defined based on ITU-T G.7710 establish a focused and efficient framework for network performance monitoring by specifying three core collection types: counts, snapshot, and tidemarks.

This intentional limitation supports key objectives such as implementation simplicity, operational efficiency, and cross-vendor interoperability. It emphasizes real-time network monitoring, favoring instantaneous or interval-based metrics over complex statistical calculations. The counts and snapshot collection types provide immediate operational data without incurring the processing overhead associated with metrics like averages and variances. These statistical measures require significant aggregation logic, which can vary across implementations and devices. By keeping computation within network elements minimal, the approach reduces both processing and memory overhead, maintaining lightweight implementations. It establishes a clear separation between raw data collection (handled by network elements) and deeper analysis (delegated to external management systems). This separation not only simplifies device requirements but also enables more consistent and flexible analytics in centralized systems, which are better equipped to apply standardized analytical frameworks.

Limiting collection types also contributes to energy efficiency by reducing the operational burden on Network Elements (NEs), while offloading data analysis to external management applications. Despite the simplicity, the selected collection types offer sufficient expressiveness to support comprehensive performance monitoring without excessive resource use. They are specifically optimized for the NE-to-client interface -- referred to as the Southbound Interface (SBI) from a controller perspective (e.g., between a Physical Network Controller (PNC) and an NE), or equivalently as the northbound management interface from the NE's perspective -- to ensure as follows:

* Lightweight to implement

* Consistently supported across vendors

* Efficient for transport and storage in network management systems

The collection types are applicable to a wide range of monitored objects, including both network topology elements (e.g., links, tunnels) and physical equipment parameters (e.g., temperature, voltage).

3.2.1. Counts

Counts measurement in network performance monitoring tracks the cumulative occurrences of specific events over a defined measurement interval, such as 15 minutes or 24 hours. This method captures how frequently certain network activities, like errors or transmission issues, occur, providing a historical view of recurring problems. Counts reset at the end of each interval, ensuring that every period starts with a fresh count for accurate monitoring.

The primary purpose of counts is to identify trends and patterns in network behavior over time, helping operators detect anomalies or areas where issues frequently arise. This type of measurement is particularly useful for long-term analysis, enabling preventive maintenance and optimizing network performance. Unlike instantaneous measurements, counts focus on aggregation over time, making it easier to understand the persistence or recurrence of faults. The data gathered through counts helps in fault management and planning by highlighting repeated errors, congestion, or performance degradation that may affect service delivery. As a result, counts provide network operators with actionable insights for troubleshooting and capacity planning, ensuring smooth operation and reliability across the network.

3.2.2. Snapshot

Snapshot is an instantaneous measurement taken at a specific point in time. It captures the instantaneous value of specific performance parameters at a regular, predefined point (uniform time) within each time interval. Snapshot provides a "momentary view" of network conditions, allowing operators to observe the network's status at specific intervals. The data from these uniform-time snapshots is then aggregated and analyzed to understand the immediate state across the entire network. By taking snapshots simultaneously across all network elements, operators can correlate data between different parts of the transport network. Snapshots are collected at pre-determined uniform times within fixed measurement intervals. The uniform time and fixed intervals can be configured based on the needs of the network.

3.2.3. Tidemarks

Tidemarks measurements record the maximum (high tidemarks) and minimum (low tidemarks) values that a performance parameter reaches during a specified measurement interval. This approach captures the extreme values and performance fluctuations, highlighting the best and worst conditions that occur within the monitoring period. Tidemarks measurements provide deeper insights by capturing performance spikes or drops that may go unnoticed in average or cumulative data, enabling precise troubleshooting of intermittent or extreme conditions. For instance, while the average error rate over a period may appear acceptable, a high tidemark could reveal intermittent spikes in errors that require attention. Conversely, a low tidemark may expose periods of severely degraded signal quality or throughput.

4. Thresholding

4.1. Periodic Thresholding

Periodic threshold events are triggered when the counts or gauge value reaches a pre-defined threshold during periodic measurements including counts, snapshot, and tidemarks for performance parameters.

The counts measurement has two types of threshold reporting methods: transient and standing condition methods. The transient condition method treats each measurement period separately. As soon as a threshold is reached or crossed in a measurement interval for a given performance measurement, a threshold report (TR) is generated. The standing condition method is optional. The standing condition is raised, and a TR (Threshold Report) is generated, when the set threshold is reached or crossed. The standing condition is cleared,

and a reset threshold report (RTR) is generated at the end of the period when the current value is below or equal to the reset threshold, provided that there was no unavailable time during that period.

For gauge measurements ("snapshot" and "tidemarks"), an overflow condition is determined and an out-of-range report is generated as soon as the gauge value reaches or crosses the high threshold. An underflow condition is determined and an out-of-range report is generated as soon as the gauge value is at or below the low threshold.

4.2. Non-Periodic Thresholding

Non-periodic threshold events are triggered regardless of the collection types (counts, snapshot, or tidemarks). The following parameters are used for non-periodic events.

* BUT (Begin Unavailable Time): The event marking the start of a period when a network element or connection is unavailable.

* EUT (End Unavailable Time): The event marking the end of a period when a network element or connection was unavailable.

* CSES (Consecutive Severely Errored Seconds): A sequence of severely errored seconds (SES) detected consecutively within a specified time interval. The reporting metrics include BUT, EUT, and the count of errors during that period.

5. Subscriptions

Using YANG-Push subscriptions [RFC8639] [RFC8640] [RFC8641], clients can receive streams of PM parameter values (PM data) produced by the counts, snapshot, and tidemarks collection types over various time intervals. This document does not define any new subscription mechanism; the examples in this section use only the standard YANG-Push RPCs and notifications. The streaming data delivered through these subscriptions is used for maintenance and Quality of Service (QoS) monitoring in networks.

Below are practical use cases demonstrating different subscription scenarios. These examples illustrate periodic and non-periodic subscriptions, including notifications triggered by threshold breaches. Each example aligns with IETF YANG-Push protocols, showcasing how network elements generate and stream performance data based on subscription parameters.

5.1. Periodic Events

The YANG-Push subscription model, as defined in [RFC8641], enables clients to subscribe to periodic performance measurement data from network elements. This model supports dynamic subscription establishment, modification, and termination for real-time streaming of PM data. Clients can specify subscription parameters including the target datastore (operational), encoding format (XML/JSON), and filtering criteria to receive only relevant performance metrics. The subscription mechanism allows for configurable update periods, enabling both high-frequency monitoring and long-term trend analysis (e.g., 24-hour intervals). Network elements generate periodic event notifications containing the requested PM data, which clients can process for real-time monitoring, historical analysis, or triggering automated responses based on performance thresholds.

Figure 5 shows a subscription request for the ES parameter in the itu-transport-maintenance-15min profile. It requests counts measurement data sampled every second and aggregated over a 15-minute interval. The reporting period is set to 900 seconds, so a notification is sent at the end of each measurement interval.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
     xmlns:sn=
       "urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications"
     xmlns:pm-coll=
       "urn:ietf:params:xml:ns:yang:ietf-pm-collection"
     message-id="101">
  <sn:establish-subscription>
    <sn:stream>YANG-PUSH</sn:stream>
    <sn:encoding>encode-xml</sn:encoding>
    <sn:filter>
      <sn:datastore>operational</sn:datastore>
      <sn:xpath-filter>
        /pm-coll:pm-periodic-measurement/
          parameter-profile[name='itu-transport-maintenance-15min']/
          pm-parameter[name='es']/
          sampling-interval[id='1s']/
          measurement-interval[id='15min']/
          collection-types/counts/measurement-value
      </sn:xpath-filter>
    </sn:filter>
    <sn:period>900</sn:period>
    <sn:anchor-time>2024-07-01T00:00:00Z</sn:anchor-time>
  </sn:establish-subscription>
</rpc>
```

Figure 5: Periodic Event Subscription Example

Figure 6 shows a notification for the ES parameter in the itu-transport-maintenance-15min profile. It reports the counts measurement value sampled every second and aggregated over a 15-minute interval. The measured value (10) represents the total errored seconds in that period.

```
<notification
    xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0"
    xmlns:pm-coll=
      "urn:ietf:params:xml:ns:yang:ietf-pm-collection">
  <eventTime>2024-07-01T00:15:00Z</eventTime>
  <pm-coll:pm-periodic-measurement>
    <parameter-profile>
      <name>itu-transport-maintenance-15min</name>
      <pm-parameter>
        <name>es</name>
        <sampling-interval>
          <id>1s</id>
          <interval-value>1</interval-value>
          <unit>second</unit>
          <measurement-interval>
            <id>15min</id>
            <interval-value>15</interval-value>
            <unit>minute</unit>
            <collection-types>
              <counts>
                <measurement-value>10</measurement-value>
              </counts>
            </collection-types>
          </measurement-interval>
        </sampling-interval>
      </pm-parameter>
    </parameter-profile>
  </pm-coll:pm-periodic-measurement>
</notification>
```

Figure 6: Periodic Event Notification Example

5.2. Threshold Events

Threshold event subscriptions enable clients to receive immediate notifications when performance metrics cross predefined thresholds, providing proactive monitoring capabilities. This subscription type uses standard YANG-Push [RFC8639] [RFC8641] datastore change notifications to deliver the threshold events defined in this document.

5.2.1. Periodic Threshold Events

```
   +--ro counts-transient
   |  +--ro event-type?       enumeration
   |  +--ro event-occurred?   boolean
   |  +--ro event-time?       yang:date-and-time
   +--ro counts-standing
   |  +--ro event-type?       enumeration
   |  +--ro event-occurred?   boolean
   |  +--ro event-time?       yang:date-and-time
   +--ro snapshot
   |  +--ro event-type?       enumeration
   |  +--ro event-occurred?   boolean
   |  +--ro event-time?       yang:date-and-time
   +--ro tidemarks
      +--ro event-type?       enumeration
      +--ro event-occurred?   boolean
      +--ro event-time?       yang:date-and-time
```

Figure 7: Periodic Threshold Events Subtree

When a performance parameter exceeds or falls below configured thresholds for the periodic collection types of counts, snapshot, and tidemarks, the network element generates event-driven notifications containing the threshold crossing event type and occurrence time (parameter values at the time of the event can be read from the operational datastore if needed). This mechanism supports four types of threshold events: count-transient-event for immediate threshold crossings, count-standing-event for persistent threshold violations, snapshot-event for instantaneous value threshold crossings, and tidemark-event for extreme value threshold crossings. These events enable rapid response to network performance degradation and automated fault management. The threshold event subscription complements periodic subscriptions by providing real-time alerts for critical performance issues that require immediate attention.

Figure 8 shows an example of the NETCONF request to subscribe to all pm-threshold-events notifications in the ietf-pm-collection model.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
     xmlns:sn=
       "urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications"
     xmlns:pm-coll=
       "urn:ietf:params:xml:ns:yang:ietf-pm-collection"
     message-id="202">
  <sn:establish-subscription>
    <sn:stream>YANG-PUSH</sn:stream>
    <sn:encoding>encode-xml</sn:encoding>
    <sn:filter>
      <sn:datastore>operational</sn:datastore>
      <sn:xpath-filter>
        /pm-coll:pm-threshold-events
      </sn:xpath-filter>
    </sn:filter>
    <sn:period>1</sn:period>
    <sn:anchor-time>2024-07-01T00:00:00Z</sn:anchor-time>
  </sn:establish-subscription>
</rpc>
```

Figure 8: Threshold Event Subscription Example

Figure 9 reports a high-OOR-event threshold crossing for the snapshot measurement of the ES parameter in the itu-transport-maintenance-15min profile, with 1-second sampling and 15-minute measurement interval. It shows the event type, occurrence, and timestamp as defined in the YANG model.

```
<notification
    xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0"
    xmlns:pm-coll=
      "urn:ietf:params:xml:ns:yang:ietf-pm-collection">
  <eventTime>2024-07-01T00:05:23Z</eventTime>
  <pm-coll:pm-threshold-events>
    <periodic-events>
      <parameter-profile>
        <name>itu-transport-maintenance-15min</name>
        <pm-parameter>
          <name>es</name>
          <sampling-interval>
            <id>1s</id>
            <interval-value>1</interval-value>
            <unit>second</unit>
            <measurement-interval>
              <id>15min</id>
              <interval-value>15</interval-value>
              <unit>minute</unit>
              <event-types>
                <snapshot>
                  <event-type>High-OOR-event</event-type>
                  <event-occurred>true</event-occurred>
                  <event-time>2024-07-01T00:05:23Z</event-time>
                </snapshot>
              </event-types>
            </measurement-interval>
          </sampling-interval>
        </pm-parameter>
      </parameter-profile>
    </periodic-events>
  </pm-coll:pm-threshold-events>
</notification>
```

Figure 9: Threshold Event Notification Example

5.2.2. Non-Periodic Threshold Events

```
   +--ro non-periodic-events
      +--ro BUT-event
      |  +--ro event-occurred?   boolean
      |  +--ro event-time?       yang:date-and-time
      +--ro EUT-event
      |  +--ro event-occurred?   boolean
      |  +--ro event-time?       yang:date-and-time
      |  +--ro duration?         uint32
      +--ro CSES-event
         +--ro event-occurred?   boolean
         +--ro start?            yang:date-and-time
         +--ro end?              yang:date-and-time
         +--ro duration?         uint32
         +--ro error-count?      uint32
```

Figure 10: Non-Periodic Threshold Events Subtree

Non-periodic threshold event subscriptions provide immediate notifications for critical network availability and error conditions that occur independently of regular measurement intervals. These subscriptions monitor for specific events such as BUT, EUT, and CSES that indicate significant network performance degradation or service interruptions. When these events occur, the network element generates immediate notifications containing event details, timing information, and duration data. This subscription type enables proactive network management by providing real-time awareness of critical network conditions that require immediate operator attention or automated intervention. Non-periodic threshold events complement periodic monitoring by capturing exceptional conditions that may not be detected through regular interval-based measurements.

6. YANG Data Model

The YANG module for PM measurements is defined below:

```
<CODE BEGINS> file "ietf-pm-collection@2026-05-02.yang"
module ietf-pm-collection {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-pm-collection";
  prefix pm-coll;

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

  organization
```

```
      "IETF Common Control and Measurement Plane (ccamp)
       Working Group";
    contact
      "WG Web: <https://datatracker.ietf.org/wg/ccamp/>
       Editor: Bin Yeong Yoon <mailto:byyun@etri.re.kr>";
    description
      "This YANG module defines a data model for performance
       management (PM) streaming from network equipment to clients,
       based on ITU-T G.7710. It supports real-time streaming of
       performance measurements using three core collection types:
       counts (cumulative events), snapshot (instantaneous values),
       and tidemarks (extreme values), as defined in ITU-T G.7710.

       The module enables proactive network monitoring through
       configurable sampling and measurement intervals, supporting
       both high-frequency real-time monitoring and long-term trend
       analysis. It provides threshold event notifications for both
       periodic measurements and non-periodic events (BUT, EUT,
       CSES).

       The design supports AI-driven applications, network digital
       twins, and dynamic network environments by enabling multiple
       simultaneous views of the same performance parameter with
       different temporal resolutions. This hierarchical structure
       allows operators, analytics systems, and digital twin
       platforms to access performance data at appropriate
       granularities while maintaining operational efficiency and
       cross-vendor interoperability.

       The module integrates with IETF YANG-Push protocols for
       subscription-based streaming, enabling clients to receive
       continuous performance data and threshold event notifications
       for real-time monitoring, historical analysis, and automated
       network management.";

    revision 2026-05-02 {
      description
        "Renamed the module from ietf-pm-measurements to
         ietf-pm-collection (including namespace and prefix).

         Terminology and model alignment: prior wording and the
         measurement-methods container for counts, snapshot, and
         tidemarks were replaced by collection types and the
         collection-types container, per ITU-T G.7710 collection-
         type concepts.";
      reference
        "RFC XXXX: A YANG Data Model of Performance Management
         Streaming";
```

```
  }

  /*
   * TYPEDEFs
   */
  typedef profile-names {
    type string {
      pattern '[a-zA-Z][a-zA-Z0-9_-]*-[a-zA-Z][a-zA-Z0-9_-]*-'
            + '[a-zA-Z][a-zA-Z0-9_-]*(-[a-zA-Z][a-zA-Z0-9_-]*)?';
    }
    description
      "Parameter profile name following the format:
       <source>-<network>-<purpose>[-<characteristic>]

       Where:
       - <source>: Standards body or organization
         (e.g., itu, ieee, ietf)
       - <network>: Network domain or layer
         (e.g., transport, access, core)
       - <purpose>: Intended use or function
         (e.g., maintenance, qos, availability)
       - <characteristic>: Optional qualifying information
         (e.g., 15min, 24hr, high-priority)

       Examples:
       - itu-transport-maintenance-15min
       - itu-transport-maintenance-24hr
       - itu-transport-qos-24hr
       - ieee-access-availability";
  }

  typedef time-interval-unit {
    type enumeration {
      enum millisecond {
        description "Time interval in milliseconds.";
      }
      enum second {
        description "Time interval in seconds.";
      }
      enum minute {
        description "Time interval in minutes.";
      }
      enum hour {
        description "Time interval in hours.";
      }
    }
    description "Units for expressing time intervals.";
  }
```

```
     /*
      * IDENTITIES
      */
     identity periodic-events {
       description
         "Base identity for periodic event notifications.";
     }

     identity counts-transient {
       base periodic-events;
       description
         "Notification for transient threshold events in counts
          measurements.";
     }

     identity counts-standing {
       base periodic-events;
       description
         "Notification for standing threshold events in counts
          measurements.";
     }

     identity snapshot {
       base periodic-events;
       description
         "Notification for snapshot measurement threshold events.";
     }

     identity tidemarks {
       base periodic-events;
       description
         "Notification for tidemarks measurement threshold events.";
     }

     identity non-periodic-events {
       description
         "Base identity for non-periodic event notifications.";
     }

     identity but {
       base non-periodic-events;
       description
         "Notification for Begin Unavailable Time (BUT) events.";
     }

     identity eut {
       base non-periodic-events;
       description
```

```
        "Notification for End Unavailable Time (EUT) events.";
    }

    identity cses {
      base non-periodic-events;
      description
        "Notification for Consecutive Severely Errored Seconds
         (CSES) events.";
    }

    /*
     * COMMON GROUPINGS
     */
    grouping threshold-config {
      description
        "Common threshold configuration for snapshot and tidemarks
         measurement types (high and low thresholds).";
      leaf high-threshold {
        type uint32;
        description
          "High threshold that triggers alerts when exceeded.";
      }
      leaf low-threshold {
        type uint32;
        description
          "Low threshold that triggers alerts when performance
           falls below acceptable levels.";
      }
    }

    grouping transient-threshold-config {
      description
        "Threshold configuration for transient conditions.
         Transient thresholds only support high threshold crossings
         and report immediately when the count value reaches or
         crosses the configured transient threshold value.
         Transient thresholds do not support low threshold
         (underflow) conditions, unlike snapshot and tidemarks
         measurements which support both high and low thresholds.";
      leaf transient-threshold {
        type uint32;
        description
          "Transient threshold that triggers alerts when exceeded.
           Transient thresholds report immediately when the count
           value reaches or crosses this threshold value.";
      }
    }
```

```
     grouping event-state-info {
       description
         "Common event state information for all event types.";
       leaf event-occurred {
         type boolean;
         description
           "Indicates whether a threshold crossing or performance
            event has occurred.";
       }
       leaf event-time {
         type yang:date-and-time;
         description
           "Precise timestamp of when the event occurred.";
       }
     }

     grouping oor-event-type {
       description
         "Common out-of-range event type definition.";
       leaf event-type {
         type enumeration {
           enum High-OOR-event {
             description "High OOR threshold exceeded.";
           }
           enum Low-OOR-event {
             description "Low OOR threshold crossed.";
           }
         }
         description
           "Specifies whether the high or low OOR threshold was
            crossed.";
       }
     }

     grouping triggered-oor-event-info {
       description
         "Combined threshold event type and event information.";
       uses oor-event-type;
       uses event-state-info;
     }

     grouping count-transient-event-type {
       description
         "Transient threshold event type definition for counts
          measurements. Transient thresholds report immediately when
          the count value reaches or crosses a configured threshold
          value.";
       leaf event-type {
```

```
        type enumeration {
          enum Threshold-Crossed-Event {
            description
              "Threshold crossing event generated when count value
               reaches or crosses the configured threshold value.";
          }
        }
        description
          "Specifies that a threshold crossing event occurred.";
      }
    }

    grouping triggered-count-transient-event-info {
      description
        "Combined transient threshold event type and event
         information for counts measurements. Transient thresholds
         are independent threshold mechanisms that report
         immediately when count values cross configured threshold
         values.";
      uses count-transient-event-type;
      uses event-state-info;
    }

    grouping time-interval-config {
      description "Common time interval configuration.";
      leaf interval-value {
        type uint32;
        description "Numeric value for the interval.";
      }
      leaf unit {
        type time-interval-unit;
        description "Time unit for the interval value.";
      }
    }

    /*
     * COLLECTION TYPE GROUPINGS
     */
    grouping count-measurement-gr {
      description
        "Counts measurement for cumulative event tracking over a
         measurement interval. Supports transient and standing
         threshold reporting, as defined in G.7710.";
      container counts {
        description
          "Contains counts measurement values and configuration.";
        container transient-condition-config {
          description
```

```
            "Configuration for transient threshold conditions.
             Transient thresholds report immediately when the
             count value reaches or crosses the configured
             transient threshold value. Transient thresholds do
             not support low threshold (underflow) conditions,
             unlike snapshot and tidemarks measurements which
             support both high and low thresholds.";
          uses transient-threshold-config;
        }
        container standing-condition-config {
          description
            "Configuration for standing condition monitoring.
             When both thresholds are set, standing-threshold
             must be greater than or equal to reset-threshold
             (hysteresis).";
          must "not(standing-threshold and reset-threshold) or "
             + "standing-threshold >= reset-threshold" {
            error-message
              "Standing threshold must be >= reset threshold.";
          }
          leaf standing-threshold {
            type uint32;
            description
              "Threshold value that triggers standing condition
               alerts.";
          }
          leaf reset-threshold {
            type uint32;
            description
              "Reset threshold value that clears standing
               conditions.";
          }
        }
        leaf measurement-value {
          type uint32;
          config false;
          description
            "Current cumulative count value for the measurement
             interval.";
        }
      }
    }

    grouping snapshot-measurement-gr {
      description
        "Snapshot measurements for instantaneous values at uniform
         time within each measurement interval. Supports high/low
         OOR threshold reporting, as defined in G.7710.";
```

```
        container snapshot {
          description
            "Contains snapshot measurement configuration and values.";
          container uniform-time-config {
            description
              "Configuration for uniform time intervals between
               snapshots.";
            leaf interval-value {
              type uint32;
              default 1;
              description
                "Numeric value for the sampling interval between
                 snapshots.";
            }
            leaf unit {
              type time-interval-unit;
              description
                "Time unit for the snapshot sampling interval.";
            }
          }
          container threshold-config {
            description
              "Configuration for snapshot threshold monitoring.";
            uses threshold-config;
          }
          leaf measurement-value {
            type uint32;
            config false;
            description
              "Current instantaneous snapshot value.";
          }
        }
      }

      grouping tidemarks-measurement-gr {
        description
          "Tidemarks measurements for maximum and minimum values
           over the measurement interval. Supports high/low OOR
           threshold reporting, as defined in G.7710.";
        container tidemarks {
          description
            "Contains tidemarks measurement values and threshold
             configuration.";
          container threshold-config {
            description
              "Configuration for tidemarks threshold monitoring.";
            uses threshold-config;
          }
```

```
        leaf high-measurement-value {
          type uint32;
          config false;
          description
            "Current maximum value recorded during the measurement
             interval.";
        }
        leaf low-measurement-value {
          type uint32;
          config false;
          description
            "Current minimum value recorded during the measurement
             interval.";
        }
      }
    }

    grouping collection-types-gr {
      description
        "Grouping for the three core collection types (counts,
         snapshot, tidemarks) per ITU-T G.7710.";
      container collection-types {
        description
          "Container for the counts, snapshot, and tidemarks
           collection types.";
        uses count-measurement-gr;
        uses snapshot-measurement-gr;
        uses tidemarks-measurement-gr;
      }
    }

    /*
     * EVENT GROUPINGS
     */
    grouping counts-transient-event-gr {
      description
        "Transient threshold events for counts measurements.
         Transient thresholds are independent threshold mechanisms
         that report immediately when count values cross configured
         threshold values.";
      container counts-transient {
        description
          "Contains information about transient threshold events
           for counts.";
        uses triggered-count-transient-event-info;
      }
    }
```

```
    grouping counts-standing-event-gr {
      description
        "Standing condition events for counts measurements.";
      container counts-standing {
        description
          "Contains information about standing threshold events
           for counts.";
        leaf event-type {
          type enumeration {
            enum Threshold-Report {
              description
                "Threshold Report (TR) generated when the count
                 value reaches or exceeds the standing-threshold
                 configured in standing-condition-config.";
            }
            enum Reset-Threshold-Report {
              description
                "Reset Threshold Report (RTR) generated at the end
                 of the period when the count value is at or below
                 the reset-threshold (G.7710 standing condition
                 clear).";
            }
          }
          description
            "Specifies the type of standing threshold event that
             occurred, as defined in G.7710. A Threshold-Report
             (TR) is generated when the measurement-value reaches
             or exceeds the standing-threshold. A
             Reset-Threshold-Report (RTR) is generated at the end
             of the period when the measurement-value is at or
             below the reset-threshold.";
        }
        uses event-state-info;
      }
    }

    grouping snapshot-events-gr {
      description
        "Threshold events for snapshot measurements.";
      container snapshot {
        description
          "Contains snapshot threshold event information.";
        uses triggered-oor-event-info;
      }
    }

    grouping tidemarks-events-gr {
      description
```

```
        "Threshold events for tidemarks measurements.";
      container tidemarks {
        description
          "Contains tidemarks threshold event information.";
        uses triggered-oor-event-info;
      }
    }

    /*
     * MEASUREMENT INTERVAL STRUCTURES
     */
    grouping periodic-measurement-intervals {
      description
        "Hierarchical structure for periodic measurement timing
         and collection types.";
      list sampling-interval {
        key "id";
        description
          "List of sampling intervals defining data collection
           frequency.";
        leaf id {
          type string;
          description
            "Unique identifier for this sampling interval
             configuration.";
        }
        leaf interval-value {
          type uint32;
          default 1;
          description "Numeric value for the sampling interval.";
        }
        leaf unit {
          type time-interval-unit;
          default second;
          description "Time unit for the sampling interval value.";
        }
        list measurement-interval {
          key "id";
          description
            "List of measurement intervals defining aggregation
             periods.";
          leaf id {
            type string;
            description
              "Unique identifier for this measurement interval
               configuration.";
          }
          leaf interval-value {
```

```
            type uint32;
            default 15;
            description
              "Numeric value for the measurement interval.";
          }
          leaf unit {
            type time-interval-unit;
            default minute;
            description
              "Time unit for the measurement interval value.";
          }
          uses collection-types-gr;
        }
      }
    }

    grouping non-periodic-events-gr {
      description
        "Grouping for non-periodic performance event parameters
         (BUT, EUT, CSES).";
      container BUT-event {
        description "Begin Unavailable Time (BUT) event.";
        uses event-state-info;
      }
      container EUT-event {
        description "End Unavailable Time (EUT) event.";
        uses event-state-info;
        leaf duration {
          type uint32;
          units "seconds";
          description
            "Total duration of unavailability in seconds.";
        }
      }
      container CSES-event {
        description
          "Consecutive Severely Errored Seconds (CSES) event.";
        leaf event-occurred {
          type boolean;
          description
            "Indicates whether a CSES event was generated.";
        }
        leaf start {
          type yang:date-and-time;
          description
            "Timestamp indicating when the CSES period began.";
        }
        leaf end {
```

```
          type yang:date-and-time;
          description
            "Timestamp indicating when the CSES period ended.";
        }
        leaf duration {
          type uint32;
          units "seconds";
          description "Duration of the CSES period in seconds.";
        }
        leaf error-count {
          type uint32;
          description
            "Number of errors during the CSES period.";
        }
      }
    }

    grouping pm-periodic-measurement-gr {
      description
        "Hierarchical structure for periodic performance
         measurements.";
      list parameter-profile {
        key "name";
        description "List of performance parameter profiles.";
        leaf name {
          type profile-names;
          description "Name of the parameter profile.";
        }
        list pm-parameter {
          key "name";
          description
            "List of PM parameters within the parameter profile.";
          leaf name {
            type string;
            description
              "Name of the performance parameter being measured.";
          }
          uses periodic-measurement-intervals;
        }
      }
    }

    /*
     * MAIN CONTAINER
     */
    container pm-periodic-measurement {
      description
        "Main container for periodic performance measurements.";
```

```
    uses pm-periodic-measurement-gr;
  }

  /*
   * NOTIFICATIONS
   */
  notification pm-threshold-events {
    description
      "Notification for periodic threshold crossing events and
       non-periodic performance events (BUT, EUT, CSES).";
    container periodic-events {
      description "Container for periodic threshold events.";
      list parameter-profile {
        key "name";
        description
          "List of performance parameter profiles for event
           monitoring.";
        leaf name {
          type profile-names;
          description "Name of the parameter profile.";
        }
        list pm-parameter {
          key "name";
          description
            "List of PM parameters within the parameter
             profile.";
          leaf name {
            type string;
            description
              "Name of the performance parameter being
               monitored.";
          }
          list sampling-interval {
            key "id";
            description
              "List of sampling intervals for event monitoring.";
            leaf id {
              type string;
              description
                "Unique identifier for this sampling interval
                 configuration.";
            }
            uses time-interval-config;
            list measurement-interval {
              key "id";
              description
                "List of measurement intervals for event
                 aggregation.";
```

```
                leaf id {
                  type string;
                  description
                    "Unique identifier for this measurement
                     interval configuration.";
                }
                uses time-interval-config;
                container event-types {
                  description
                    "Container for different threshold event
                     types.";
                  uses counts-transient-event-gr;
                  uses counts-standing-event-gr;
                  uses snapshot-events-gr;
                  uses tidemarks-events-gr;
                }
              }
            }
          }
        }
      }

      container non-periodic-events {
        description
          "Container for non-periodic performance events (BUT,
           EUT, CSES).";
        uses non-periodic-events-gr;
      }
    }
  }
  <CODE ENDS>
```

7. YANG Data Trees

```
  module: ietf-pm-collection
    +--rw pm-periodic-measurement
       +--rw parameter-profile* [name]
          +--rw name              profile-names
          +--rw pm-parameter* [name]
             +--rw name                 string
             +--rw sampling-interval* [id]
                +--rw id                  string
                +--rw interval-value?     uint32
                +--rw unit?               time-interval-unit
                +--rw measurement-interval* [id]
                   +--rw id               string
                   +--rw interval-value? uint32
                   +--rw unit?           time-interval-unit
```

```
                  +--rw collection-types
                     +--rw counts
                     |  +--rw transient-condition-config
                     |  |  +--rw transient-threshold?   uint32
                     |  +--rw standing-condition-config
                     |  |  +--rw standing-threshold?    uint32
                     |  |  +--rw reset-threshold?       uint32
                     |  +--ro measurement-value?        uint32
                     +--rw snapshot
                     |  +--rw uniform-time-config
                     |  |  +--rw interval-value?   uint32
                     |  |  +--rw unit?             time-interval-unit
                     |  +--rw threshold-config
                     |  |  +--rw high-threshold?   uint32
                     |  |  +--rw low-threshold?    uint32
                     |  +--ro measurement-value?   uint32
                     +--rw tidemarks
                        +--rw threshold-config
                        |  +--rw high-threshold?   uint32
                        |  +--rw low-threshold?    uint32
                        +--ro high-measurement-value?   uint32
                        +--ro low-measurement-value?    uint32

notifications:
  +---n pm-threshold-events
     +--ro periodic-events
     |  +--ro parameter-profile* [name]
     |     +--ro name              profile-names
     |     +--ro pm-parameter* [name]
     |        +--ro name                 string
     |        +--ro sampling-interval* [id]
     |           +--ro id                string
     |           +--ro interval-value?   uint32
     |           +--ro unit?             time-interval-unit
     |           +--ro measurement-interval* [id]
     |              +--ro id             string
     |              +--ro interval-value? uint32
     |              +--ro unit?          time-interval-unit
     |              +--ro event-types
     |                 +--ro counts-transient
     |                 |  +--ro event-type?      enumeration
     |                 |  +--ro event-occurred?  boolean
     |                 |  +--ro event-time?      yang:date-and-time
     |                 +--ro counts-standing
     |                 |  +--ro event-type?      enumeration
     |                 |  +--ro event-occurred?  boolean
     |                 |  +--ro event-time?      yang:date-and-time
     |                 +--ro snapshot
```

```
|                 |  +--ro event-type?      enumeration
|                 |  +--ro event-occurred?  boolean
|                 |  +--ro event-time?      yang:date-and-time
|                 +--ro tidemarks
|                    +--ro event-type?      enumeration
|                    +--ro event-occurred?  boolean
|                    +--ro event-time?      yang:date-and-time
+--ro non-periodic-events
   +--ro BUT-event
   |  +--ro event-occurred?   boolean
   |  +--ro event-time?       yang:date-and-time
   +--ro EUT-event
   |  +--ro event-occurred?   boolean
   |  +--ro event-time?       yang:date-and-time
   |  +--ro duration?         uint32
   +--ro CSES-event
      +--ro event-occurred?   boolean
      +--ro start?            yang:date-and-time
      +--ro end?              yang:date-and-time
      +--ro duration?         uint32
      +--ro error-count?      uint32
```

Figure 11: Tree of pm measurements module

8. PM Interval Capabilities

The ietf-pm-interval-capabilities YANG module provides comprehensive capability discovery for interval configurations, enabling clients to understand the supported temporal resolutions before configuring PM measurements. This module is designed to work seamlessly with the ietf-pm-collection module, supporting both real-time monitoring scenarios requiring high-frequency sampling and long-term trend analysis requiring extended measurement intervals.

The advertisement of interval capabilities follows standard IETF procedures for capability discovery in network management systems, based on the principles outlined in [RFC9195] for YANG instance data files, [RFC8525] for YANG library, and [RFC9196] for system and notification capability advertisement.

8.1. Motivation

ITU-T G.7710 does not currently include a clause that explicitly mandates a capability-discovery mechanism for configurable measurement timing parameters. However, the configurability of sampling and measurement intervals defined in ITU-T G.7710, together with the streaming model defined therein, requires that clients know in advance which intervals a given server can support, so that

subscription requests can be constructed without violating implementation constraints. This module provides that capability advertisement.

The need for interval capabilities discovery arises from several critical factors that affect the configuration and operation of performance monitoring systems. Different types of performance parameters inherently require different sampling and measurement intervals based on their characteristics and measurement objectives. For example, error-based parameters such as errored seconds (ES) may require frequent sampling to capture transient events, while availability metrics might be adequately monitored with longer intervals. Similarly, latency measurements often need high-frequency sampling to detect microsecond-level variations, whereas throughput statistics can be effectively captured with less frequent sampling.

Vendor dependencies represent another significant factor necessitating interval capabilities discovery. Network equipment manufacturers implement different hardware architectures, processing capabilities, and measurement engines, resulting in varying support for sampling and measurement intervals. Some vendors may support very fine-grained intervals (e.g., millisecond-level sampling) for high-precision applications, while others may be optimized for longer intervals suitable for operational monitoring. Additionally, different vendors may have different constraints on the relationship between sampling and measurement intervals, with some supporting only specific multiples or ranges.

The complexity of modern network environments further underscores the importance of interval capabilities discovery. Multi-vendor networks require clients to adapt their monitoring strategies based on the specific capabilities of each network element. Without proper capability discovery, clients risk configuring unsupported intervals, leading to configuration failures, suboptimal monitoring, or even system instability. The interval capabilities framework addresses these challenges by providing a standardized mechanism for discovering and understanding the temporal resolution capabilities of network elements, enabling clients to make informed decisions about interval configuration and ensuring interoperability across diverse network environments.

The interval capabilities module follows a hierarchical structure that mirrors the measurement configuration model, ensuring consistency between capability discovery and actual measurement configuration. The architecture consists of three levels: Parameter Profiles (collections of related performance parameters such as itu-transport-maintenance-15min), PM Parameters (individual performance parameters within profiles such as es, ses, bbe), and Interval Capabilities (sampling and measurement interval capabilities for each parameter).

The module defines a critical relationship between sampling and measurement intervals: measurement intervals must be multiples of their corresponding sampling intervals. This constraint ensures that measurement aggregation periods align with data collection frequency, preventing configuration errors and ensuring accurate performance monitoring. For example, if a device supports a 5-second sampling interval, valid measurement intervals would be 5s, 10s, 15s, 30s, 60s, etc. This relationship is enforced through the hierarchical structure where measurement intervals are defined within their corresponding sampling intervals.

8.2. Capability Discovery and Configuration Workflow

A NETCONF client can discover the sampling and measurement interval capabilities of a server by following the standard IETF procedures for capability and module discovery. This process involves multiple steps, beginning with the session establishment and extending to operational data retrieval.

Upon initiating a session, the client receives the server’s <hello> message as defined in [RFC6241]. This message includes a list of capability URIs, indicating the supported YANG modules and protocol extensions. If the server includes entries for both ietf-pm-collection and ietf-pm-interval-capabilities, the client infers that the server supports performance measurement with parameter-specific intervals, and also advertises its supported interval values.

To confirm module support and retrieve metadata such as revision dates and feature availability, the client queries the YANG Library as defined in [RFC8525]. This step allows the client to verify that both the measurement model and the interval capability model are implemented and discoverable.

The client then queries the pm-interval-capabilities container, which is defined with config false and thus resides in the operational datastore per the Network Management Datastore Architecture (NMDA) described in [RFC8342]. By querying this container, the client can retrieve a list of supported sampling and measurement intervals for

each performance parameter and profile. The structure includes constraints such as minimum and maximum values, allowed time units, and granularity. This live runtime exposure of capability information follows the model described in [RFC9196] for advertising telemetry and notification capabilities in operational state.

Alternatively, the same interval capabilities may be published as static data files using the format defined in [RFC9195]. This allows vendors or standards bodies to document the supported measurement intervals of a device or profile without requiring a live connection to the system.

At this point, the client uses the retrieved interval capabilities to configure a performance measurement subscription using the pm-collection model. This configuration is made in alignment with the update intervals supported by the server, ensuring compatibility and preventing errors such as period-unsupported, as outlined in [RFC8641].

This end-to-end process ensures that performance measurement configurations are both valid and optimized for the server’s capabilities, leveraging both static publication and runtime introspection using standardized models and procedures.

8.3. Interval Capabilities Example

The following example demonstrates how a client can discover interval capabilities for the ES parameter in the itu-transport-maintenance-15min profile, specifically requesting 1-second sampling with 15-minute measurement intervals.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
     xmlns:pm-int-cap=
       "urn:ietf:params:xml:ns:yang:ietf-pm-interval-capabilities"
     message-id="301">
  <get>
    <filter>
      <pm-int-cap:pm-interval-capabilities>
        <parameter-profile>
          <name>itu-transport-maintenance-15min</name>
          <pm-parameter>
            <name>es</name>
            <interval-relationships>
              <sampling-interval>
                <id>1s</id>
                <min-value>1</min-value>
                <max-value>1</max-value>
                <units>second</units>
                <default-value>1</default-value>
                <default-unit>second</default-unit>
                <granularity>1</granularity>
                <measurement-interval>
                  <id>measurement-range</id>
                  <min-value>5</min-value>
                  <max-value>1440</max-value>
                  <units>minute</units>
                  <default-value>15</default-value>
                  <default-unit>minute</default-unit>
                  <granularity>5</granularity>
                </measurement-interval>
              </sampling-interval>
            </interval-relationships>
          </pm-parameter>
        </parameter-profile>
      </pm-int-cap:pm-interval-capabilities>
    </filter>
  </get>
</rpc>
```

Figure 12: Interval Capabilities Discovery Example

This example shows how a client can discover that the network element supports 1-second sampling with a flexible measurement interval range (5-1440 minutes) with 5-minute granularity. The response confirms unit support for seconds in sampling and minutes in measurement intervals, with a default recommendation of 15 minutes, enabling the client to choose any appropriate measurement duration within the supported range for their PM monitoring requirements.

8.4. YANG Data Model

```
<CODE BEGINS> file "ietf-pm-interval-capabilities@2026-05-02.yang"
module ietf-pm-interval-capabilities {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-pm-interval-capabilities";
  prefix ipc;

  import ietf-pm-collection {
    prefix pm-coll;
    reference "draft-ietf-ccamp-pm-measurements-yang";
  }

  organization
    "IETF Common Control and Measurement Plane (ccamp)
     Working Group";
  contact
    "WG Web: <https://datatracker.ietf.org/wg/ccamp/>
     Editor: Bin Yeong Yoon <mailto:byyun@etri.re.kr>";
  description
    "This YANG module defines a data model for discovering and
     expressing performance management (PM) interval capabilities
     in network equipment. The module enables clients to discover
     what sampling and measurement intervals a server can support
     for different performance parameters within various
     parameter profiles.

     This module provides hierarchical interval capability
     discovery where measurement intervals must be multiples of
     their corresponding sampling intervals, and each parameter
     can have different interval capabilities within parameter
     profiles.

     This module is designed to work with ietf-pm-collection for
     complete PM streaming solutions and supports both real-time
     monitoring and long-term trend analysis.";

  revision 2026-05-02 {
    description
      "Companion module ietf-pm-collection replaces measurement
       methods with collection types (counts, snapshot,
       tidemarks) and renames the grouping and container to
       collection-types per ITU-T G.7710; this module continues
       to describe interval capabilities for that design.";
    reference
      "RFC XXXX: A YANG Data Model of Performance Management
       Streaming";
```

```
  }

  /*
   * TYPEDEFs
   */
  typedef interval-unit {
    type enumeration {
      enum millisecond {
        description "Time interval in milliseconds.";
      }
      enum second {
        description "Time interval in seconds.";
      }
      enum minute {
        description "Time interval in minutes.";
      }
      enum hour {
        description "Time interval in hours.";
      }
      enum day {
        description "Time interval in days.";
      }
    }
    description "Supported units for expressing time intervals.";
  }

  /*
   * IDENTITIES
   */
  identity interval-capability-type {
    description
      "Base identity for different types of interval
       capabilities.";
  }

  identity sampling-interval-capability {
    base interval-capability-type;
    description
      "Capability for sampling intervals - how frequently data
       is collected.";
  }

  identity measurement-interval-capability {
    base interval-capability-type;
    description
      "Capability for measurement intervals - how long
       measurements are aggregated.";
  }
```

```
    /*
     * GROUPINGS
     */
    grouping interval-constraints {
      description
        "Constraints for supported intervals including min/max
         values and supported units. This grouping defines the
         capability constraints for both sampling and measurement
         intervals, allowing devices to express their supported
         interval ranges, units, and granularity.";
      leaf min-value {
        type uint32;
        description
          "Minimum supported value for this interval type.";
      }
      leaf max-value {
        type uint32;
        description
          "Maximum supported value for this interval type.";
      }
      leaf-list units {
        type interval-unit;
        description
          "List of supported time units for this interval type.";
      }
      leaf default-value {
        type uint32;
        description
          "Default value recommended for this interval type.";
      }
      leaf default-unit {
        type interval-unit;
        description
          "Default unit recommended for this interval type.";
      }
      leaf granularity {
        type uint32;
        description
          "Granularity step for interval values. For example, if
           granularity is 5, then only values that are multiples
           of 5 are supported.";
      }
    }

    grouping parameter-interval-capabilities {
      description
        "Interval capabilities for a specific parameter within a
         profile. This grouping defines the complete interval
```

```
          capability information for a single performance
          parameter, including its name and hierarchical interval
          relationships. The structure uses mapping-based discovery
          where all interval information is contained within the
          sampling-interval structure.";
        leaf name {
          type string;
          description
            "Name of the performance parameter (e.g., es, ses,
             bbe).";
        }
        container interval-relationships {
          description
            "Defines hierarchical relationships between sampling and
             measurement intervals for this parameter. Measurement
             intervals must be multiples of their corresponding
             sampling intervals.";
          list sampling-interval {
            key "id";
            description
              "Maps sampling intervals to their compatible
               measurement intervals with complete capability
               information.";
            leaf id {
              type string;
              description
                "Unique identifier for this sampling interval
                 capability.";
            }
            uses interval-constraints;
            list measurement-interval {
              key "id";
              description
                "Detailed information for each compatible
                 measurement interval within the sampling interval
                 structure.";
              leaf id {
                type string;
                description
                  "Unique identifier for this measurement interval
                   capability.";
              }
              uses interval-constraints;
            }
          }
        }
      }
```

```
    grouping profile-parameter-capabilities {
      description
        "Capabilities for all parameters within a specific
         parameter profile. This grouping defines the complete
         capability information for a parameter profile, including
         the profile name and all performance parameters with
         their respective interval capabilities.";
      leaf name {
        type pm-coll:profile-names;
        description
          "Name of the parameter profile (e.g.,
           itu-transport-maintenance-15min).";
      }
      list pm-parameter {
        key "name";
        description
          "List of parameters with their specific interval
           capabilities within this profile.";
        uses parameter-interval-capabilities;
      }
    }

    /*
     * MAIN CONTAINER
     */
    container pm-interval-capabilities {
      description
        "Main container for hierarchical PM interval capabilities.
         This container provides comprehensive information about
         the sampling and measurement intervals that a server can
         support for different parameters within different
         parameter profiles.";
      config false;

      list parameter-profile {
        key "name";
        description
          "List of parameter profiles with their parameter-
           specific interval capabilities. Each profile represents
           a collection of parameters that share common
           measurement requirements but may have different
           interval capabilities.";
        uses profile-parameter-capabilities;
      }
    }
  }
  <CODE ENDS>
```

8.5. YANG Data Trees

```
module: ietf-pm-interval-capabilities
  +--ro pm-interval-capabilities
     +--ro parameter-profile* [name]
        +--ro name              pm-coll:profile-names
        +--ro pm-parameter* [name]
           +--ro name                    string
           +--ro interval-relationships
              +--ro sampling-interval* [id]
                 +--ro id              string
                 +--ro min-value?      uint32
                 +--ro max-value?      uint32
                 +--ro units*          interval-unit
                 +--ro default-value?  uint32
                 +--ro default-unit?   interval-unit
                 +--ro granularity?    uint32
                 +--ro measurement-interval* [id]
                    +--ro id              string
                    +--ro min-value?      uint32
                    +--ro max-value?      uint32
                    +--ro units*          interval-unit
                    +--ro default-value?  uint32
                    +--ro default-unit?   interval-unit
                    +--ro granularity?    uint32
```

Figure 13: Tree of pm interval capabilities module

9. Manageability Considerations

This section will be completed in a future revision of this document. Considerations to be addressed include the operational impact of large numbers of concurrent YANG-Push subscriptions for PM data, alignment of measurement intervals with NE clock sources, and interaction with existing fault and configuration management workflows.

10. Security Considerations

This section will be completed in a future revision of this document. The YANG modules defined in this document define data nodes that are designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF. The lowest NETCONF layer is the secure transport layer, and the mandatory- to-implement secure transport is Secure Shell (SSH). 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.

Detailed analysis of sensitive readable nodes, writable nodes, and RPC operations will be added in a future revision.

11. IANA Considerations

This document requests IANA to register the following URIs in the "ns" subregistry within the "IETF XML Registry" [RFC3688]:

URI: urn:ietf:params:xml:ns:yang:ietf-pm-collection Registrant Contact: The IESG. XML: N/A; the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-pm-interval-capabilities Registrant Contact: The IESG. XML: N/A; the requested URI is an XML namespace.

This document also requests IANA to register the following YANG modules in the "YANG Module Names" registry [RFC6020]:

Name: ietf-pm-collection Namespace: urn:ietf:params:xml:ns:yang:ietf-pm-collection Prefix: pm-coll Reference: RFC XXXX

Name: ietf-pm-interval-capabilities Namespace: urn:ietf:params:xml:ns:yang:ietf-pm-interval-capabilities Prefix: ipc Reference: RFC XXXX

12. References

12.1. Normative References

[G7710] ITU-T, "Common Equipment Management Function Requirements", ITU-T Recommendation G.7710, November 2025, <https://www.itu.int/rec/T-REC-G.7710>.

[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>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/rfc/rfc8342>.

[RFC8525] Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", RFC 8525, DOI 10.17487/RFC8525, March 2019, <https://www.rfc-editor.org/rfc/rfc8525>.

[RFC8639] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Subscription to YANG Notifications", RFC 8639, DOI 10.17487/RFC8639, September 2019, <https://www.rfc-editor.org/rfc/rfc8639>.

[RFC8640] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Dynamic Subscription to YANG Events and Datastores over NETCONF", RFC 8640, DOI 10.17487/RFC8640, September 2019, <https://www.rfc-editor.org/rfc/rfc8640>.

[RFC8641] Clemm, A. and E. Voit, "Subscription to YANG Notifications for Datastore Updates", RFC 8641, DOI 10.17487/RFC8641, September 2019, <https://www.rfc-editor.org/rfc/rfc8641>.

[RFC9196] Lengyel, B., Clemm, A., and B. Claise, "YANG Modules Describing Capabilities for Systems and Datastore Update Notifications", RFC 9196, DOI 10.17487/RFC9196, February 2022, <https://www.rfc-editor.org/rfc/rfc9196>.

12.2. Informative References

[RFC3593] Tesink, K., Ed., "Textual Conventions for MIB Modules Using Performance History Based on 15 Minute Intervals", RFC 3593, DOI 10.17487/RFC3593, September 2003, <https://www.rfc-editor.org/rfc/rfc3593>.

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

[RFC3705] Ray, B. and R. Abbi, "High Capacity Textual Conventions for MIB Modules Using Performance History Based on 15 Minute Intervals", RFC 3705, DOI 10.17487/RFC3705, February 2004, <https://www.rfc-editor.org/rfc/rfc3705>.

[RFC4656] Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., and M. Zekauskas, "A One-way Active Measurement Protocol (OWAMP)", RFC 4656, DOI 10.17487/RFC4656, September 2006, <https://www.rfc-editor.org/rfc/rfc4656>.

[RFC5357] Hedayat, K., Krzanowski, R., Morton, A., Yum, K., and J. Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)", RFC 5357, DOI 10.17487/RFC5357, October 2008, <https://www.rfc-editor.org/rfc/rfc5357>.

[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>.

[RFC8762] Mirsky, G., Jun, G., Nydell, H., and R. Foote, "Simple Two-Way Active Measurement Protocol", RFC 8762, DOI 10.17487/RFC8762, March 2020, <https://www.rfc-editor.org/rfc/rfc8762>.

[RFC9195] Lengyel, B. and B. Claise, "A File Format for YANG Instance Data", RFC 9195, DOI 10.17487/RFC9195, February 2022, <https://www.rfc-editor.org/rfc/rfc9195>.

[RFC9197] Brockners, F., Ed., Bhandari, S., Ed., and T. Mizrahi, Ed., "Data Fields for In Situ Operations, Administration, and Maintenance (IOAM)", RFC 9197, DOI 10.17487/RFC9197, May 2022, <https://www.rfc-editor.org/rfc/rfc9197>.

Contributors

Kwangkoog Lee
KT
Email: kwangkoog.lee@kt.com

Jongyoon Shin
SK Telecom
Email: jongyoon.shin@sk.com

Sungyong Nam
LGU+
Email: sy.nam@lguplus.co.kr

Authors' Addresses

Bin Yeong Yoon
ETRI
Email: byyun@etri.re.kr

Youngkil You
woori-net
Email: young@woori-net.com