CCAMP Working Group
Internet-Draft
Intended status: Standards Track
Expires: 20 April 2026

B. Y. Yoon
ETRI

17 October 2025

A YANG Data Model of Performance Management Streaming
draft-yoon-ccamp-pm-streaming-03

Abstract

This document provides a YANG data model of performance management streaming from network equipment to clients. It defines PM measurement methods, event notifications, generic PM parameters and streaming subscriptions. Additionally, it includes a YANG module for PM interval capabilities discovery, enabling clients to understand supported sampling and measurement intervals before configuring PM measurements.

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 20 April 2026.

Table of Contents

1. Introduction

With the rise of AI-driven applications, network digital twins, and increasingly dynamic network environments, there is growing demand for performance management (PM) streaming capabilities. PM streaming enables proactive issue detection, allowing network operators to identify and address potential problems before they affect service. It also helps optimize resource allocation, ensuring efficient use of bandwidth and other network resources.

[ITU-T_G7710] provides a foundational framework for managing transport network elements, addressing requirements, parameters, and measurement methods for performance management. However, [ITU-T_G7710] does not define YANG data models or specific protocols needed for PM streaming, which are essential for modern network management. To support PM streaming, various IETF documents and protocols [RFC8639], [RFC8640], [RFC8641] can be utilized. This document provides a YANG data model for PM streaming in network equipment based on [ITU-T_G7710], demonstrating how to subscribe to the YANG model using the IETF push model.

2. PM Streaming

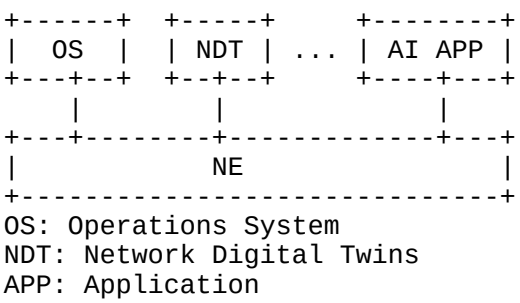

Figure 1: Streaming Network Architecture

PM streaming is a real-time method for measuring and transmitting data to monitor the performance and health of network devices and systems. It provides valuable insights into key metrics like errored seconds (ES), latency, and packet loss, helping to optimize networks, detect anomalies, and manage faults proactively. Unlike traditional periodic data collection, PM streaming delivers continuous updates, enabling faster, more responsive network adjustments.

Using telemetry protocols like YANG Push, PM streaming allows for more frequent and detailed performance monitoring. By integrating this data into AI-driven analytics, it supports preemptive interventions, enhancing overall network reliability. Additionally, it keeps digital twins synchronized with the physical network, offering real-time insights for predictive maintenance, planning, and optimization.

The procedures for Performance Management streaming between a network node and clients such as operations system (OS), AI applications, network digital twins (NDT) involve continuous measurement of performance metrics on PM parameters using three methods: counts (tracking event occurrences), snapshot (instantaneous metric values), and tidemarks (extreme values over a period). Clients can initiate the process by sending a subscription request specifying the metrics, measurement methods, intervals, and filtering criteria. Once the node confirms the subscription, it collects and aggregates PM data based on the requested metrics and intervals. Notifications with PM data, including timestamps, metrics, and measurement methods, are sent to clients at each interval via protocols like NETCONF or RESTCONF. Clients then process the data, using it for real-time monitoring, historical analysis, or triggering alerts based on thresholds. They can also manage subscriptions by modifying parameters or suspending them as needed.

3. PM parameters

3.1. 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_G7710], 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.

3.2. 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 an identity derived from the parameter-profile-name base identity. 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 ietf-qos-ip-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_G7710], 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 identities, 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.

3.2.1. 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, 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
* ietf-ip-qos-24hr
* vendorx-otn-sla

'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.

3.3. Transport Common PM Parameters

Metric value of PM parameters is 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

* UAS Unavailable Seconds

According to the types of the measurement methods, 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:

3.3.1. 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 measurement methods (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.

3.3.2. 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 measurement methods (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.

3.3.3. 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 measurement method 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 measurement method 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.

3.3.4. 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.

4. Periodic Measurement

4.1. 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 measurement methods 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.

4.1.1. Use Cases

The hierarchical design of the ietf-pm-measurements YANG module, wherein a performance parameter supports multiple sampling intervals and each sampling interval may be associated with multiple measurement intervals and methods, is motivated by a set of operationally validated use cases as shown in the following table. In these use cases, a single logical client-whether a human operator, network application, or analytics engine-requires simultaneous access to multiple views of the same performance parameter, differentiated by temporal resolution and analysis objective.

In network operations centers (NOCs), an operator may require performance monitoring based on high-frequency samples (e.g., 1-second sampling with 15-minute measurement intervals), while concurrently assessing longer-term service-level agreement (SLA) compliance through longer measurement windows (e.g., 24-hour aggregation). It aligns with the performance monitoring functions and applications on performance monitoring parameters such as errored seconds (ES), etc. described in [ITU-T_G7710].

A single dashboard or planning tool may correlate short-term utilization spikes with long-term trends using different sampling and measurement combinations on the same parameter. Network Performance Monitor for Critical Link Stability monitor tracks latency using the counts method at multiple time resolutions to address immediate service degradation and long-term path reliability. A single monitoring platform benefits from consistent sampling while leveraging different measurement intervals to inform short-term action and long-term optimization.

An AI/ML analytics system may ingest the same parameter, such as ES at different granularities for multiple purposes: high-resolution data for anomaly detection, medium-resolution tidemarks for real-time model updates, and low-resolution tidemarks for daily or monthly training. All of these operations may be performed within the scope of a single analytics application.

A digital twin platform continuously mirrors the real-time behavior of a physical network using packet delay variation (pdv). To accurately represent dynamic conditions, the system uses very fine-grained sampling with multiple count-based measurement intervals to feed simulation models and predictive engines in real time. Measurement data is ingested at short intervals (e.g., 1 minute) to maintain near-instantaneous synchronization with the physical network’s current performance, supporting accurate digital mirroring. A slightly longer interval (e.g., 5 minutes) allows the twin to detect and buffer small fluctuations, supporting feedback loops that

smooth reactive behavior and adapt to transient changes. Long-term intervals (e.g., 1 hour) feed higher-order analytics and AI-based learning models that detect drift, optimize resource allocation, and improve future simulation fidelity.

In all these cases, the hierarchical list structure-where parameters contain multiple sampling intervals, and each sampling interval defines one or more measurement intervals and methods-supports operational flexibility, avoids configuration duplication, and enables fine-grained control of measurement strategies.

Client Type (parameter)	Sampling Interval	Measurement Interval	Measurement Methods	Purpose
NOC (ES)	1s	15min	tidemarks	Maintenance
		24hr	tidemarks	QoS
NOC (latency)	500ms	1min	counts	Delay spikes
		30min	counts	Recurring bursts
		24hr	counts	Trends
AI/ML Analytics System (ES)	1s	1min	tidemarks	Anomaly detect
		15min	tidemarks	Trend
		24hr	tidemarks	Model training
Digital Twin (pdv)	100ms	1min	counts	Synchronization
		5min	counts	Feedback loop
		1hr	counts	Learning model

Figure 4: Use cases of sampling and measurement intervals

4.2. Measurement Methods

```
   +--rw measurement-methods
      +--rw counts
      |  +--rw transient-condition-config
      |  |  +--rw high-threshold?   uint32
      |  |  +--rw low-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 5: Measurement Methods Subtree

The measurement methods defined based on [ITU-T_G7710] establish a focused and efficient framework for network performance monitoring by specifying three core methods: 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. Counts and snapshot methods provide immediate operational data without incurring the processing overhead associated with metrics like averages and variances etc. 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 measurement methods 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 measurement methods offer sufficient expressiveness to support comprehensive performance monitoring without excessive resource use. So, they are specifically optimized for Southbound Interface (SBI) between Physical Network Controllers (PNCs) and NEs to ensures as follows:

* Lightweight to implement

* Consistently supported across vendors

* Efficient for transport and storage in network management systems

The measurement methods 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).

4.2.1. 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.

4.2.2. 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.

4.2.3. 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 tidemarks could reveal intermittent spikes in errors that require attention. Conversely, a low tidemarks may expose periods of severely degraded signal quality or throughput.

5. Thresholding

5.1. 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 an 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.

5.2. Non-Periodic Thresholding

Non-periodic threshold events are triggered regardless of the measurement methods (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.

6. Subscriptions

Clients can receive a streaming of values of the PM parameters (PM data) by the measurement methods of counts, snapshot, and tidemarks with various time intervals, after subscribing to them in equipment. The streaming data measured on the PM parameters are 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.

6.1. 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 6 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-meas="urn:ietf:params:xml:ns:yang:ietf-pm-measurements"
     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-meas:pm-periodic-measurement/
          parameter-profile[name='itu-transport-maintenance-15min']/
          pm-parameter[name='es']/
          sampling-interval[id='1s']/
          measurement-interval[id='15min']/
          measurement-methods/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 6: Periodic Event Subscription Example

Figure 7 This XML 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-meas="urn:ietf:params:xml:ns:yang:ietf-pm-measurements">
  <eventTime>2024-07-01T00:15:00Z</eventTime>
  <pm-meas: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>
            <measurement-methods>
              <counts>
                <measurement-value>10</measurement-value>
              </counts>
            </measurement-methods>
          </measurement-interval>
        </sampling-interval>
      </pm-parameter>
    </parameter-profile>
  </pm-meas:pm-periodic-measurement>
</notification>
```

Figure 7: Periodic Event Notification Example

6.2. Threshold Events

Threshold event subscriptions enable clients to receive immediate notifications when performance metrics cross predefined thresholds, providing proactive monitoring capabilities. This subscription type, based on [RFC8639] YANG Push for datastore change notifications, allows clients to define threshold conditions.

6.2.1. 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 8: Periodic Threshold Events Subtree

When a performance parameter exceeds or falls below configured thresholds for the periodic measurement methods of counts, snapshot, and tidemarks, the network element generates event-driven notifications containing detailed information about the threshold crossing event, including event type, occurrence time, and current parameter values. 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 9 shows an example of the NETCONF request to subscribe to all pm-threshold-events notifications in the ietf-pm-measurements 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-meas="urn:ietf:params:xml:ns:yang:ietf-pm-measurements"
     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-meas: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 9: Threshold Event Subscription Example

Figure 10 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-meas="urn:ietf:params:xml:ns:yang:ietf-pm-measurements">
  <eventTime>2024-07-01T00:05:23Z</eventTime>
  <pm-meas: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-meas:pm-threshold-events>
</notification>
```

Figure 10: Threshold Event Notification Example

6.2.2. 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 11: 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.

7. YANG Data Model

The YANG module for PM measurements is defined below:

```
module ietf-pm-measurements {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-pm-measurements";
  prefix pm-meas;

  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 comprehensive 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
        methods: counts (cumulative events), snapshot (instantaneous
        values), and tidemarks (extreme values).

        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 2025-10-17 {
       description
         "Initial version.";
       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 all measurement types.";
      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 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 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.";
      }
    }

    /*
     * MEASUREMENT METHOD GROUPINGS
     */
    grouping count-measurement-gr {
      description
        "Counts measurement for cumulative event tracking.";
      container counts {
        description
          "Contains counts measurement values and configuration.";
        container transient-condition-config {
          description
            "Configuration for transient out-of-range conditions.";
          uses threshold-config;
        }
        container standing-condition-config {
          description
            "Configuration for standing condition monitoring.";
          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.";
      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 extreme values.";
      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 measurement-methods-gr {
      description
        "Container for the three core measurement methods.";
      container measurement-methods {
        description
          "Container for different measurement methods.";
        uses count-measurement-gr;
        uses snapshot-measurement-gr;
        uses tidemarks-measurement-gr;
      }
    }

    /*
     * EVENT GROUPINGS
     */
```

```
    grouping counts-transient-event-gr {
      description
        "Threshold events for counts measurements.";
      container counts-transient {
        description
          "Contains information about transient threshold events for
           counts.";
        uses triggered-oor-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.";
            }
            enum Reset-Threshold-Report {
              description "Reset Threshold Report (RTR) generated.";
            }
          }
          description
            "Specifies whether a TR or RTR was generated.";
        }
        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
         methods.";
      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 measurement-methods-gr;
        }
      }
    }

    grouping non-periodic-events-gr {
      description
        "Container for non-periodic event parameters.";
      container BUT-event {
        description "Begin Unavailable Time (BUT) event.";
        uses event-state-info;
      }
      container EUT-event {
        description "End Unavailable Time (EUT) event.";
        leaf event-occurred {
          type boolean;
          description
            "Indicates whether an EUT event was generated.";
        }
        leaf event-time {
          type yang:date-and-time;
          description "Precise timestamp of the EUT event.";
        }
        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 threshold crossing events.";
      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 threshold events.";
         uses non-periodic-events-gr;
       }
     }
   }
```

8. YANG Data Trees

```
  module: ietf-pm-measurements
    +--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 measurement-methods
                      +--rw counts
```

```
                      |  +--rw transient-condition-config
                      |  |  +--rw high-threshold?   uint32
                      |  |  +--rw low-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 12: Tree of pm measurements module

9. 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-measurements 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 module capability advertisement, [RFC8525] for YANG library, and [RFC9196] for YANG module advertisement.

9.1. Motivation

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.

9.2. 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-measurements 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-measurements 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.

9.3. 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 13: 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.

9.4. YANG Data Model

```
   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-measurements {
       prefix pm-meas;
       reference "draft-yoon-ccamp-pm-streaming-03";
     }

     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-measurements
        for complete PM streaming solutions and supports both
        real-time monitoring and long-term trend analysis.";

     revision 2025-10-17 {
       description
         "Initial version.";
       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, including hierarchical interval relationships
           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-meas: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;
      }

    }

  }
```

9.5. YANG Data Trees

```
module: ietf-pm-interval-capabilities
  +--ro pm-interval-capabilities
     +--ro parameter-profile* [name]
        +--ro name            pm-meas: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 14: Tree of pm interval capabilities module

10. Manageability Considerations

<Add any manageability considerations>

11. Security Considerations

<Add any security considerations>

12. IANA Considerations

<Add any IANA considerations>

13. References

13.1. Normative References

[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>.

[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>.

[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>.

13.2. Informative References

[ITU-T_G7710] ITU-T, "Common Equipment Management Function Requirements", November 2022, <https://www.itu.int/rec/T-REC-G.7710>.

Author’s Address

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