Workgroup: CCAMP Working Group
Internet-Draft: draft-yoon-ccamp-pm-streaming-01
Published: 2 April 2025
Intended Status: Standards Track
Expires: 4 October 2025
Author: B.Y. Yoon, Ed.
ETRI

A YANG Data Model of Performance Management Streaming

Abstract

This document provides a YANG data model of performance management streaming in network equipment. It defines types of parameters, and their measurements and reporting methods by periodic and event notifications.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 3
2. Terminology 3
3. PM Streaming 3
4. Measurement Methods 4
  4.1. Counts 4
  4.2. Snapshots 5
  4.3. Tidemarks 5
5. Periodic Subscriptions 5
  5.1. Maintenance and QoS monitoring 5
  5.2. Time intervals 6
  5.3. PM Parameters 7
6. Threshold Event Subscriptions 8
  6.1. Periodic threshold events 8
  6.2. Non-Periodic threshold events 8
7. YANG Data Model 9
  7.1. PM measurements module 9
  7.2. PM parameters module 17
  7.3. Relationship between measurement and parameter modules 19
8. YANG Data Tree 19
9. Subscription Examples 21
  9.1. Periodic Subscription 21
  9.2. Periodic Event Subscription 23
10. Security Considerations 24
11. IANA Considerations 24
12. References 24
  12.1. Normative References 24
  12.2. Informative References 25

Author's Address 25

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.

The ITU-T G.7710 standard provides a foundational framework for managing transport network elements, addressing requirements, parameters, and measurement methods for performance management. However, G.7710 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 ([RFC9232], [RFC8639], [RFC8640], [RFC8641]) can be utilized. This document provides a YANG data model for PM streaming in network equipment based on ITU-T G.7710, demonstrating how to subscribe to the YANG model using the IETF push model.

2. Terminology

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

3. PM Streaming

```
          +-------+  +-----+     +---------+
          |  OS   |  | NDT | ... | AI APP |
          +---+---+  +--+--+     +----+---+
              |         |             |
          +---+--------+-------------+---+
          |             NE               |
          +------------------------------+
          OS: Operation System
          NDT: Network Digital Twins
          APP: Application
```

Figure 1: PM streaming telemetry

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 operation systems (OS), AI applications, Network digital twins (NDT) involve continuous measurement of performance metrics on PM parameters using three methods: counts (tracking event occurrences), snapshots (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.

4. Measurement Methods

In network performance monitoring, ITU-T G.7710 defines two measurement methods that measure a metric value with periodic intervals such as 15-minute or 24-hour periods: counter measurements ("counts") and gauge measurements ("snapshots" and "tidemarks"). These complementary methods provide a holistic view of network behavior by capturing both long-term trends and real-time conditions. The counter measurements provide data on the frequency of specific events, enabling long-term analysis of recurring issues. Meanwhile, the gauge measurement involves capturing instantaneous values of performance monitoring metrics at one-second intervals. This high-frequency sampling occurs during defined periodic intervals rather than continuously. They offer real-time insight and capture extreme conditions, helping to identify immediate performance problems or deviations from normal behavior.

4.1. Counts

Counts measurement in network performance monitoring tracks the cumulative occurrences of specific events over a defined period, 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. Snapshots

Snapshots are instantaneous measurements taken at a specific point in time. They capture the instantaneous value of specific performance parameters at a regular, predefined point (uniform time) within each time interval. Snapshots provide 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 intervals (e.g., every 15 minutes, 24 hours). The uniform time and fixed intervals can be configured based on the needs of the network.

4.3. Tidemarks

Tidemark measurements record the maximum (high tidemark) and minimum (low tidemark) values that a performance parameter reaches during a specified observation window. These extreme values offer insight into the range of performance fluctuations, highlighting the best and worst conditions that occur within the monitoring period. Tidemark 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.

5. Periodic Subscriptions

Clients can receive a streaming of values of the PM parameters (PM data) by the measurement methods of counts, snapshots, 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.

5.1. Maintenance and QoS monitoring

Performance management data contribute to enhancing overall network reliability. It consists of Maintenance and QoS data generated by maintenance and QoS parameters respectively. These two categories serve different purposes and focus on distinct aspects of network management.

By continuously tracking different aspects of network performance, they ensure that the network infrastructure remains robust, capable of meeting operational demands, and able to deliver consistent and high-quality services to users.

The Maintenance data is to ensure the overall operational integrity and reliability of the network. It is focused on maintaining the health of the physical network infrastructure, detecting and diagnosing faults, and addressing any issues that could compromise network stability. Maintenance-related performance parameters are typically aimed at identifying physical layer issues, equipment malfunctions, or any performance degradation. These metrics are particularly useful for preventive and corrective maintenance activities. Common maintenance metrics might include error counts, signal degradation measurements, and transmission errors. These types of data help in triggering alarms or initiating repairs to physical network elements, such as optical fiber problems or electrical signal faults.

On the other hand, the QoS data is primarily concerned with monitoring and ensuring the quality of services delivered over the network. While maintenance focuses on network health at the infrastructure level, QoS is centered on the performance of data transmission and how effectively the network meets the needs of end-users. QoS data is used to measure the quality of services such as voice, video, and other data applications that require stable and reliable network performance. Metrics for QoS often include latency (the delay in data transmission), jitter (the variation in packet arrival times), packet loss, bandwidth utilization, and throughput. These metrics are crucial in ensuring that the network delivers services at a level that meets or exceeds the expectations outlined in service-level agreements (SLAs).

Maintenance data typically deal with the health and status of individual network elements or links. They often involve unidirectional measurements to detect faults or issues in specific components or paths. QoS data often need to be measured in both directions of a communication path to ensure overall service quality. This is because many applications and services rely on two-way communication, and the user experience depends on performance in both directions.

5.2. Time intervals

In network performance management, periodic time intervals of 15 minutes and 24 hours are commonly used for network nodes to measure and notify performance data of clients. These intervals serve distinct purposes, helping network operators monitor both short-term fluctuations and long-term trends in network performance.

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.

In the context of emerging technologies like AI and network digital twins, smaller sampling intervals are becoming increasingly necessary. These technologies demand real-time or near-real-time performance data to function optimally. As these capabilities become more integral to advanced network operations, the shift towards smaller sampling intervals ensures that they can operate efficiently, react to network changes faster, and improve overall network reliability and performance.

5.3. 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.

- Maintenance parameters measured with counts, snapshots and tidemarks over 15-minute time interval: ES, SES, BBE, BBC, UAS
- Extended maintenance parameters measured with counts, snapshots and tidemarks over 24-hour time interval: ES, SES, BBE, BBC, UAS, PJE
- QoS parameters measured with counts over 24-hour time interval: ES, SES, BBE, BBC, SEP, UAS

6. Threshold Event Subscriptions

Threshold Events are triggered when a metric value reaches or crosses a pre-defined threshold. There are two types of threshold event notifications: periodic and non-periodic event notifications.

6.1. Periodic threshold events

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

[Counter measurement events]

Counter measurement ("counts") 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 15-minute/24-hour period for a given performance measurement, a threshold report (TR) is generated. The standing condition method is an option for 15-minute periods. 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.

[Gauge measurement events]

For gauge measurements ("snapshots" 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. These out-of-range methods are applicable for 15-minute and 24-hour measurements.

6.2. Non-Periodic threshold events

Non-periodic events are triggered regardless of the measurement methods (counts, snapshots, 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.

7. YANG Data Model

7.1. PM measurements module

The pm-measurements module consists of a periodic measurement container and notification of threshold events. The pm-periodic-measurement container aggregates various measurement types (counts, snapshots, tidemarks) and collects data at regular intervals. It includes mechanisms for continuous count-based tracking, instantaneous snapshot captures, and the recording of peak values over time. This integrated approach enables operators to obtain a complete and detailed view of network performance trends, allowing them to monitor steady behavior, capture transient conditions at specific moments, and identify peak usage or stress periods effectively.

Complementing the periodic measurements, the threshold events notification (periodic-threshold-events and non-periodic-threshold-events) incorporates an event detection mechanism that uses threshold criteria to trigger alerts when performance metrics deviate from expected ranges. One aspect focuses on scheduled comparisons where measurements are consistently evaluated against predefined thresholds, while the other captures unexpected or isolated deviations outside of the regular monitoring cycle. This dual-layered strategy not only helps in identifying persistent issues, but also ensures that sudden, anomalous performance events are promptly recognized and addressed.

```
<CODE BEGINS> file "ietf-pm-measurements@2025-04-02.yang"

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

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

  import ietf-pm-parameters {
    prefix pm-param;
    reference
      "draft-ietf-ccamp-pm-parameters-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 performance
      management measurements, based on ITU-T G.7710 for
      equipment. It specifies measurement types, performance
      parameters, their measurement methods, and event
      notifications.";

   revision 2025-04-02 {
     description
       "Initial revision for pm measurements.";
   }

   /*
    * TYPEDEFs
    */
   typedef parameter-unit {
     type string;
     description
       "Unit of measurement for performance management
        parameters defined by network elements (servers).";
   }

   typedef parameter-name {
     type union {
       type pm-param:maintenance-parameters;
       type pm-param:maintenance-parameters-extended;
       type pm-param:qos-parameters;
       type string {
         pattern
           '^([a-zA-Z][a-zA-Z0-9\-_]*:)?[a-zA-Z][a-zA-Z0-9\-_]*$';
       }
     }
     description
       "Name of the performance parameter to be measured.
        Can be either a parameter from ietf-pm-parameters module
        or a custom parameter in the format [module-name:]parameter-name.";
   }

   /*
    * GROUPINGS
    */
   grouping out-of-range-config {
     description
       "An out-of-range (OOR) event is triggered to indicate
        an overflow or underflow condition when a measured
        metric exceeds a configured high threshold or falls below a
        configured low threshold. These events apply to count (transient),
        snapshot, and tidemark measurements.";
     leaf high-oor-threshold {
       type uint32;
       description
         "High out-of-range threshold.";
     }
     leaf low-oor-threshold {
       type uint32;
       description
         "Low out-of-range threshold.";
     }
   }

   grouping event-state-info {
     leaf event-occurred {
       type boolean;
       description
         "Indicates whether an event has occurred.";
     }
     leaf event-time {
       type yang:date-and-time;
       description
         "Timestamp of when the event occurred.";
     }
   }

   grouping oor-event-type {
     leaf event-type {
       type enumeration {
         enum High-OOR-event {
           description
             "High OOR threshold reached";
         }
         enum Low-OOR-event {
           description
             "Low OOR threshold reached";
         }
       }
       description
         "Indicates whether the high or low OOR threshold was reached.";
     }
   }

   grouping triggered-oor-event-info {
     uses oor-event-type;
     uses event-state-info;
     description
       "Combines threshold event type and event details for a triggered OOR
event.";
   }

   grouping count-measurement-gr {
     description
       "Counts are cumulative measurements tracking the total occurrences of
        specific network events or activities over a time interval (e.g., 15
minutes).
        Counts are reset at regular intervals to start new measurements.";
     container count-measurement {
       leaf count-value {
         type uint32;
         config false;
       }
       leaf count-unit {
         type parameter-unit;
         config false;
         description
```

```
            "Unit of measurement for the count-value.";
        }
        container transient-condition-config {
          uses out-of-range-config;
        }
        container standing-condition-config {
          description
            "Configuration of the standing condition
             method for count measurements.";
          leaf standing-threshold {
            type uint32;
            description
              "Standing threshold value.";
          }
          leaf standing-reset-threshold {
            type uint32;
            description
              "Reset threshold value.";
          }
        }
      }
    }

    grouping snapshot-measurement-gr {
      container snapshot-measurement {
        description
          "Snapshots are instantaneous measurements taken at specific points
           in time, providing a momentary view of network conditions. Snapshots
           are collected at uniform intervals (e.g., 15 minutes or 24 hours).";
        leaf uniform-time {
          type uint32;
          units "seconds";
          description
            "Interval between snapshot measurements.";
        }
        leaf snapshot-value {
          type uint32;
          config false;
        }
        leaf snapshot-unit {
          type parameter-unit;
          config false;
          description
            "Unit of measurement for the snapshot-value.";
        }
        uses out-of-range-config;
      }
    }

    grouping tidemark-measurement-gr {
      description
        "Tidemarks record the highest (maximum) and lowest (minimum) values
         of a specific performance metric over a defined period (e.g., 15
  minutes
         or 24 hours). They capture extreme values reflecting significant
  performance peaks.";

      container tidemark-measurement {
        leaf high-tide-value {
          type uint32;
          config false;
        }
        leaf high-tide-unit {
          type parameter-unit;
          config false;
          description
            "Unit of measurement for the high-tide-value.";
        }
        leaf low-tide-value {
          type uint32;
          config false;
        }
        leaf low-tide-unit {
          type parameter-unit;
          config false;
          description
            "Unit of measurement for the low-tide-value.";
        }
        uses out-of-range-config;
      }
    }

    grouping count-transient-event-gr {
      description
        "A Threshold Report (TR) is generated for a counter when its value
         reaches or crosses the configured threshold.";
      container count-transient-event {
        uses triggered-oor-event-info;
      }
    }

    grouping count-standing-event-gr {
      description
        "A Threshold Report (TR) is generated when the set threshold is
  exceeded.
         A Reset Threshold Report (RTR) is issued at the end of the period if
  the
         value falls below or equals the reset threshold.";
      container count-standing-event {
        leaf event-type {
          type enumeration {
            enum Threshold-Report {
              description
                "High OOR threshold reached";
            }
            enum Reset-Threshold-Report {
              description
                "Low OOR threshold reached";
            }
          }
          description
            "Indicates whether a TR or RTR was generated.";
        }
        uses event-state-info;
      }
    }
```

```
  grouping snapshot-events-gr {
    container snapshot-event {
      description
        "High (or low) snapshot event triggered when the snapshot-value
reaches
         or crosses a high (or low) OOR threshold.";
      uses triggered-oor-event-info;
    }
  }

  grouping tidemark-events-gr {
    container tidemark-event {
      description
        "High (or low) tidemark event triggered when the tidemark-value
reaches
         or crosses a high (or low) OOR threshold.";
      uses triggered-oor-event-info;
    }
  }

  grouping event-parameters-gr {
    container BUT-event {
      description
        "Begin Unavailable Time (BUT) event marking the start of a period
when a
         network element or connection becomes unavailable.";
      uses event-state-info;
    }
    container EUT-event {
      description
        "End Unavailable Time (EUT) event marking the end of a period when a
         network element or connection was unavailable.";
      leaf event-occurred {
        type boolean;
        description
          "Indicates whether an EUT event was generated.";
      }
      leaf event-time {
        type yang:date-and-time;
        description
          "Timestamp of the EUT event.";
      }
      leaf total-unavailable-time {
        type uint32;
        units "seconds";
        description
          "Total duration of unavailability.";
      }
    }
    container CSES-event {
      description
        "Consecutive Severely Errored Seconds (CSES) event detected within a
specified time frame.";
      leaf event-occurred {
        type boolean;
        description
          "Indicates whether a CSES event was generated.";
      }
      leaf begin-time {
        type yang:date-and-time;
        description
          "Timestamp indicating when the CSES period began.";
      }
      leaf end-time {
        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.";
      }
      leaf error-count {
        type uint32;
        description
          "Number of errors during the CSES period.";
      }
    }
  }

  grouping pm-parameter-entry {
    leaf parameter-name {
      type parameter-name;
      description
        "Name of the performance parameter being measured.";
    }
  }

  grouping pm-threshold-events-gr {
    container periodic-threshold-events {
      description
        "Threshold crossing events for periodic measurements (counts,
snapshots, tidemarks).";
      list pm-parameter {
        key "parameter-name";
        description
          "List of performance parameters for which periodic threshold events
are reported.";
        uses pm-parameter-entry;
        uses count-transient-event-gr;
        uses count-standing-event-gr;
        uses snapshot-events-gr {
          refine "snapshot-event/event-occurred" {
            description
              "Indicates whether the snapshot-value reached a high or low OOR
threshold.";
          }
          refine "snapshot-event/event-time" {
            description
              "Timestamp when the snapshot-value crossed the high or low OOR
threshold.";
          }
        }
        uses tidemark-events-gr {
          refine "tidemark-event/event-occurred" {
            description
              "Indicates whether the tidemark-value reached a high or low OOR
threshold.";
          }
          refine "tidemark-event/event-time" {
            description
              "Timestamp when the tidemark-value crossed the high or low OOR
threshold.";
          }
        }
      }
    }
    container non-periodic-threshold-events {
      uses event-parameters-gr;
    }
  }

  grouping pm-periodic-measurement-gr {
    list pm-parameter {
      key "parameter-name";
      description
        "List of performance parameters being measured periodically.";
      uses pm-parameter-entry;
      uses count-measurement-gr;
      uses snapshot-measurement-gr;
      uses tidemark-measurement-gr;
    }
  }

  /*
   * MAIN CONTAINER
   */
  container pm-periodic-measurement {
    description
      "Contains periodic performance measurements (count, snapshot, tidemark)
       for maintenance and QoS monitoring. Measurements are typically taken
       over 15-minute and 24-hour intervals and categorized by purpose
(maintenance or QoS).";
    uses pm-periodic-measurement-gr;
  }

  /*
   * NOTIFICATIONS
   */
  notification pm-threshold-events {
    description
      "Threshold crossing events for performance parameters.";
    uses pm-threshold-events-gr;
  }
}

<CODE ENDS>
```

7.2. PM parameters module

The PM parameter module defines performance management parameters with 15-minute and 24-hour measurement intervals and QoS parameters with a 24-hour measurement interval. It serves as a repository for key measurement settings and operational parameters, enabling consistent configuration and interoperability across diverse network devices and management systems.

```
<CODE BEGINS> file "ietf-pm-parameters@2025-04-02.yang"

module ietf-pm-parameters {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-pm-parameters";
    prefix pm-param;

    organization
      "IETF 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 performance management parameters
       based on ITU-T G.7710.";

    revision 2025-04-02 {
        description
          "Updated revision to define parameters.";
    }

    typedef maintenance-parameters {
        type enumeration {
            enum es {
                description "Errored Second";
            }
            enum ses {
                description "Severely Errored Second";
            }
            enum bbe {
                description "Background Block Error";
            }
            enum bbc {
                description "Background Block Count";
            }
            enum uas {
                description "Unavailable Second";
            }
        }
        description
            "Enumeration of performance parameters used for
            maintenance monitoring with 15 minute time
            interval.";
    }

    typedef maintenance-parameters-extended {
        type enumeration {
            enum es {
                description "Errored Second";
            }
            enum ses {
                description "Severely Errored Second";
            }
            enum bbe {
                description "Background Block Error";
            }
            enum bbc {
                description "Background Block Count";
            }
            enum uas {
                description "Unavailable Second";
            }
            enum pje {
                description "Pointer Justification Event";
            }
        }
        description
            "Enumeration of performance parameters used for
            maintenance monitoring with 24 hour time
            interval.";
    }

    typedef qos-parameters {
        type enumeration {
            enum es {
                description "Errored Second";
            }
            enum ses {
                description "Severely Errored Second";
            }
            enum bbe {
                description "Background Block Error";
            }
            enum bbc {
                description "Background Block Count";
            }
            enum uas {
                description "Unavailable Second";
            }
            enum sep {
                description "Severely Errored Period";
            }
        }
        description
            "Enumeration of performance parameters used for
            QoS monitoring with 24 hours.";
    }

}
```

```
<CODE ENDS>
```

7.3. Relationship between measurement and parameter modules

The performance measurements module leverages the imported parameters from the PM parameters module, ensuring that all performance-related settings, such as thresholds and timing information, are defined consistently. By doing so, it promotes uniformity across the system, making sure that the data collected adheres to a common structure and can be easily interpreted by different network components.

This modular approach not only streamlines the management of measurement parameters but also facilitates interoperability between different network devices and management systems. As a result, any updates or refinements made to the parameter definitions in the PM parameters module automatically benefit the measurements module, reinforcing the overall flexibility and maintainability of the performance management framework.

8. YANG Data Tree

The YANG data tree structure is as follows:

```
module: ietf-pm-measurements
  +--rw pm-periodic-measurement
     +--rw pm-parameter* [parameter-name]
        +--rw parameter-name          parameter-name
        +--rw count-measurement
        |  +--ro count-value?                  uint32
        |  +--ro count-unit?                   parameter-unit
        |  +--rw transient-condition-config
        |  |  +--rw high-oor-threshold?   uint32
        |  |  +--rw low-oor-threshold?    uint32
        |  +--rw standing-condition-config
        |     +--rw standing-threshold?         uint32
        |     +--rw standing-reset-threshold?   uint32
        +--rw snapshot-measurement
        |  +--rw uniform-time?         uint32
        |  +--ro snapshot-value?       uint32
        |  +--ro snapshot-unit?        parameter-unit
        |  +--rw high-oor-threshold?   uint32
        |  +--rw low-oor-threshold?    uint32
        +--rw tidemark-measurement
           +--ro high-tide-value?      uint32
           +--ro high-tide-unit?       parameter-unit
           +--ro low-tide-value?       uint32
           +--ro low-tide-unit?        parameter-unit
           +--rw high-oor-threshold?   uint32
           +--rw low-oor-threshold?    uint32

  notifications:
    +---n threshold-events
       +--ro periodic-threshold-events
       |  +--ro pm-parameter* [parameter-name]
       |     +--ro parameter-name           parameter-name
       |     +--ro count-transient-event
       |     |  +--ro event-type?       enumeration
       |     |  +--ro event-occurred?   boolean
       |     |  +--ro event-time?       yang:date-and-time
       |     +--ro count-standing-event
       |     |  +--ro event-type?       enumeration
       |     |  +--ro event-occurred?   boolean
       |     |  +--ro event-time?       yang:date-and-time
       |     +--ro snapshot-event
       |     |  +--ro event-type?       enumeration
       |     |  +--ro event-occurred?   boolean
       |     |  +--ro event-time?       yang:date-and-time
       |     +--ro tidemark-event
       |        +--ro event-type?       enumeration
       |        +--ro event-occurred?   boolean
       |        +--ro event-time?       yang:date-and-time
       +--ro non-periodic-threshold-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 total-unavailable-time?   uint32
```

```
           +--ro CSES-event
              +--ro event-occurred?   boolean
              +--ro begin-time?       yang:date-and-time
              +--ro end-time?         yang:date-and-time
              +--ro duration?         uint32
              +--ro error-count?      uint32
```

Figure 2: Tree of pm measurements module

```
ietf-pm-parameters@2025-04-02
├── typedef maintenance-parameters
│   └── enum
│       ├── es (Errored Second)
│       ├── ses (Severely Errored Second)
│       ├── bbe (Background Block Error)
│       ├── bbc (Background Block Count)
│       └── uas (Unavailable Second)
│
├── typedef maintenance-parameters-extended
│   └── enum
│       ├── es (Errored Second)
│       ├── ses (Severely Errored Second)
│       ├── bbe (Background Block Error)
│       ├── bbc (Background Block Count)
│       ├── uas (Unavailable Second)
│       └── pje (Pointer Justification Event)
│
└── typedef qos-parameters
    └── enum
        ├── es (Errored Second)
        ├── ses (Severely Errored Second)
        ├── bbe (Background Block Error)
        ├── bbc (Background Block Count)
        ├── uas (Unavailable Second)
        └── sep (Severely Errored Period)
```

Figure 3: Tree of pm parameters

9. Subscription Examples

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.

9.1. Periodic Subscription

Figure 4 shows an example of the NETCONF request that subscribes to receive periodic notifications for multiple measurement methods (count, snapshot, and tidemark) of the Severely Errored Second (SES) parameter for maintenance with a 15-minute interval.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
     xmlns:sn="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications"
     xmlns:pm-param="urn:ietf:params:xml:ns:yang:ietf-pm-parameters"
     xmlns:pm-meas="urn:ietf:params:xml:ns:yang:ietf-pm-measurements"
     message-id="101">
    <establish-subscription
        xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications">
        <stream>YANG-PUSH</stream>
        <encoding>encode-xml</encoding>
        <filter>
            <datastore>operational</datastore>
            <xpath-filter>
                /pm-meas:pm-periodic-measurement/pm-parameter[
                    parameter-name='pm-param:ses'
                ]/count-measurement/count-value
            </xpath-filter>
        </filter>
        <period>900</period>
        <anchor-time>2024-03-19T00:00:00Z</anchor-time>
    </establish-subscription>
</rpc>
```

Figure 4: Periodic Subscription Example

Figure 5 shows an example of the NETCONF notification resulting from the subscription above, containing data from all three measurement methods.

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0"
              xmlns:pm-meas="urn:ietf:params:xml:ns:yang:ietf-pm-
measurements"
              xmlns:pm-param="urn:ietf:params:xml:ns:yang:ietf-pm-
parameters">
    <eventTime>2024-03-19T00:30:00Z</eventTime>
    <pm-meas:pm-periodic-measurement>
        <pm-parameter>
            <parameter-name>pm-param:ses</parameter-name>
            <count-measurement>
                <count-value>4</count-value>
                <count-unit>seconds</count-unit>
            </count-measurement>
        </pm-parameter>
    </pm-meas:pm-periodic-measurement>
</notification>
```

Figure 5: Periodic Notification Example

9.2. Periodic Event Subscription

Figure 6 shows an example of the NETCONF request to subscribe to receive event notifications triggered by threshold events for multiple measurement methods of the Severely Errored Second (SES) parameter for maintenance with a 15-minute interval. The subscription includes both periodic threshold events (for counts and snapshots) and non-periodic threshold events (for BUT, EUT, and CSES).

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
     xmlns:sn="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications"
     xmlns:pm-param="urn:ietf:params:xml:ns:yang:ietf-pm-parameters"
     xmlns:pm-meas="urn:ietf:params:xml:ns:yang:ietf-pm-measurements"
     message-id="107">
    <establish-subscription
        xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications">
        <stream>YANG-PUSH</stream>
        <encoding>encode-xml</encoding>
        <filter>
            <datastore>operational</datastore>
            <xpath-filter>
                /pm-meas:threshold-events/periodic-threshold-events/
                pm-parameter[parameter-name='pm-param:ses']/
                snapshot-event[event-type='High-OOR-event']
            </xpath-filter>
        </filter>
        <dampening-period>0</dampening-period>
    </establish-subscription>
</rpc>
```

Figure 6: Periodic Event Subscription Example

Figure 7 shows an example of the NETCONF notification that would be triggered after the subscription in Figure 6, containing both periodic and non-periodic threshold events.

```
<?xml version="1.0" encoding="UTF-8"?>
<notification
    xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0"
    xmlns:pm-meas="urn:ietf:params:xml:ns:yang:ietf-pm-measurements"
    xmlns:pm-param="urn:ietf:params:xml:ns:yang:ietf-pm-parameters">
    <eventTime>2024-03-19T00:15:00Z</eventTime>
    <pm-meas:threshold-events>
        <periodic-threshold-events>
            <pm-parameter>
                <parameter-name>pm-param:ses</parameter-name>
                <snapshot-event>
                    <event-type>High-OOR-event</event-type>
                    <event-occurred>true</event-occurred>
                    <event-time>2024-03-19T00:15:00Z</event-time>
                </snapshot-event>
            </pm-parameter>
        </periodic-threshold-events>
    </pm-meas:threshold-events>
</notification>
```

Figure 7: Periodic Event Notification Example

10. Security Considerations

TBD.

11. IANA Considerations

TBD.

12. References

12.1. Normative References

[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/info/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/info/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/info/rfc8641>.

12.2. Informative References

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

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

[RFC9232] Song, H., Qin, F., Martinez-Julia, P., Ciavaglia, L., and A. Wang, "Network Telemetry Framework", RFC 9232, DOI 10.17487/RFC9232, May 2022, <https://www.rfc-editor.org/info/rfc9232>.

Author's Address

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