Operations and Management Area Working Group
Internet-Draft
Intended status: Standards Track
Expires: 6 December 2026

J. Evans, Ed.
Amazon
O. Pylypenko, Ed.
Nvidia
J. Haas
Juniper Networks
A. Kadosh
Cisco Systems, Inc.
M. Boucadair, Ed.
Orange

4 June 2026

Information and Data Models for Packet Discard Reporting
draft-ietf-opsawg-discardmodel-13

Abstract

This document defines an Information Model and specifies a corresponding YANG data model for packet discard reporting. The Information Model provides an implementation-independent framework for classifying packet loss - both intended (e.g., due to policy) and unintended (e.g., due to congestion or errors) - to enable automated network mitigation of unintended packet loss. The YANG data model specifies an implementation of this Information Model for network elements with a focus on the interface, device, and control-plane discards.

About This Document

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

The latest revision of this draft can be found at https://o-pylypenko.github.io/draft-ietf-opsawg-discardmodel/draft-ietf-opsawg-discardmodel.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-opsawg-discardmodel/.

Discussion of this document takes place on the Operations and Management Area Working Group mailing list (mailto:opsawg@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/opsawg/. Subscribe at https://www.ietf.org/mailman/listinfo/opsawg/.

Source for this draft and an issue tracker can be found at https://github.com/o-pylypenko/draft-ietf-opsawg-discardmodel.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 3
  1.1. Editorial Note (To be removed by the RFC Editor) 4
2. Terminology 5
3. Problem Statement 5
4. Information Model (IM) 7
  4.1. Structure 7
  4.2. Subtype Definitions 10
  4.3. "ietf-packet-discard-reporting-common" YANG Module . . . 11
  4.4. "ietf-packet-discard-reporting-sx" YANG Module 25
5. Data Model (DM) 27
  5.1. Structure 28
  5.2. Implementation Requirements 30
  5.3. Usage Examples 31
  5.4. "ietf-packet-discard-reporting" YANG Module 33
6. Operational Considerations 37

6.1. Deployment Experience 37
6.2. Anchoring Flow Structure 38
7. Implementation Status 38
7.1. Information Model Implementations 38
7.2. Data Model Implementations 39
8. Security Considerations 39
8.1. Information Model 39
8.2. Data Model 39
9. IANA Considerations 40
10. References 41
10.1. Normative References 41
10.2. Informative References 42
Appendix A. Where Do Packets Get Dropped? 45
Appendix B. Example Signal-to-mitigation Action Mapping 46
Appendix C. Full Information Model Tree 47
Appendix D. Full Data Model Tree 54
Acknowledgements 58
Contributors 58
Authors' Addresses 58

1. Introduction

The primary function of a network is to transport and deliver packets according to service level objectives. For network operators, understanding both where and why packet loss occurs within a network is essential for effective operation. Device-reported packet loss provides the most direct signal for identifying service impact. While certain types of packet loss, such as policy-based discards, are intentional and part of normal network operation, unintended packet loss can impact customer services. To automate network operations, operators must be able to detect customer-impacting packet loss, determine its root cause, and apply appropriate mitigation actions. Precise classification of packet loss is thus crucial to ensure that anomalous packet loss is easily detected and that the right action is taken to mitigate the impact. Taking the wrong action can make problems worse; for example, removing a congested device from service can exacerbate congestion by redirecting traffic to other already congested links or devices.

Existing metrics for reporting packet loss, such as ifInDiscards, ifOutDiscards, ifInErrors, and ifOutErrors defined in "The Interfaces Group MIB" [RFC2863] and "A YANG Data Model for Interface Management" [RFC8343], are insufficient for automating network operations. First, they lack precision; for instance, ifInDiscards aggregates all discarded inbound packets without specifying the cause, making it challenging to distinguish between intended and unintended discards. Second, these definitions are ambiguous, leading to inconsistent vendor implementations. For example, in some implementations

ifInErrors accounts only for errored packets that are dropped, while in others, it includes all errored packets, whether they are dropped or not. Many implementations support more discard metrics than these, however, they have been inconsistently implemented due to the lack of a standardised classification scheme and clear semantics for packet loss reporting. For example, [RFC7270] provides support for reporting discards per flow in IP Flow Information Export (IPFIX) [RFC7011] using the forwardingStatus IPFIX Information Element, however, the defined drop reason codes also lack sufficient clarity to facilitate automated root cause analysis and impact mitigation (e.g., the "For us" reason code).

This document defines an Information Model (IM) and specifies a corresponding YANG Data Model (DM) for packet loss reporting to address the above issues. The IM provides precise classification of packet loss to enable accurate automated mitigation. The DM specifies a YANG implementation of this IM for network elements, while maintaining consistency through clear semantics.

The scope of this document is limited to reporting packet loss at Layer 3 and frames discarded at Layer 2. This document considers only the signals that may trigger automated mitigation actions and not how the actions are defined or executed. Such considerations are deployment-specific.

Section 3 describes the problem space and requirements. Section 4 defines the IM and its classification scheme. Section 5 specifies the corresponding YANG data model and implementation requirements together with a set of usage examples, and the complete YANG module definition. Appendices A and B provide additional context and implementation guidance.

1.1. Editorial Note (To be removed by the RFC Editor)

Note to the RFC Editor: This section is to be removed prior to publication.

This document contains placeholder values that need to be replaced with finalized values at the time of publication. This note summarizes all of the substitutions that are needed.

Please apply the following replacements:

* XXXX --> the assigned RFC number for this I-D

* 2026-03-03 --> the actual date of the publication of this document

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.

Tree diagrams used in this document follow the notation defined in [RFC8340].

This document makes use of the following terms:

Packet discard: It accounts for any instance where a packet is dropped by a device, regardless of whether the discard was intentional or unintentional.

Intended packet discards (Intended discards, for short): Are packets dropped due to deliberate network policies or configurations designed to enforce security or Quality of Service (QoS). For example, packets dropped because they match an Access Control List (ACL) denying certain traffic types.

Unintended packet discards (Unintended discards, for short): Are packets that were dropped, which the network operator otherwise intended to deliver, i.e., which indicates an error state. There are many possible reasons for unintended packet loss, including: erroring links may corrupt packets in transit; incorrect routing tables may result in packets being dropped because they do not match a valid route; configuration errors may result in a valid packet incorrectly matching an ACL and being dropped.

Device discard counters do not by themselves establish operator intent. Discards reported under policy (e.g., ACL/policer) indicate only that traffic matched a configured rule; such discards may still be unintended if the configuration is in error. Determining intent for policy discards requires external context (e.g., configuration validation and change history) which is out of scope for this specification.

3. Problem Statement

The fundamental problem for network operators is how to automatically detect when and where unintended packet loss is occurring and determine the appropriate action to mitigate it. For any network, there are a small set of potential actions that can be taken to mitigate customer impact when unintended packet loss is detected, for example:

1. Take a problematic device, link, or set of devices and/or links out of service.

2. Return a device, link, or set of devices and/or links back into service.

3. Move traffic to other links or devices to alleviate congestion or avoid problematic paths.

4. Roll back a recent change to a device that might have caused the problem.

5. Escalate to a network operator as a last resort when automated mitigation is not possible.

The ability to select the appropriate mitigation action depends on four key features of packet loss:

FEATURE-DISCARD-SCOPE: Determines which devices, interfaces, and/or flows are impacted. This also needs to cover control plane discards.

FEATURE-DISCARD-RATE: The rate and/or magnitude of the discards, indicating the severity and urgency of the problem. Rate may be expressed using absolute (e.g., packets per second (pps)) or relative (e.g., percent) values.

FEATURE-DISCARD-DURATION: The duration of the discards which helps to distinguish transient from persistent issues.

FEATURE-DISCARD-CLASS: The type or class of discards, which is crucial for selecting the appropriate type of mitigation. Examples may be: error discards may require taking faulty components out of service, no-buffer discards may require traffic redistribution, or intended policy discards typically require no action. Refer to Table 1 for more examples.

While most of FEATURE-DISCARD-SCOPE, FEATURE-DISCARD-RATE, and FEATURE-DISCARD-DURATION are implicitly supported by the Interfaces Group MIB [RFC2863] and the YANG Data Model for Interface Management [RFC8343], FEATURE-DISCARD-CLASS requires a more detailed classification scheme than they define. The IM provided in Section 4 defines such a classification scheme to enable automated mapping from discard signals to appropriate mitigation actions (refer to Appendix B for examples).

4. Information Model (IM)

The IM is defined using YANG [RFC7950], with Data Structure Extensions [RFC8791], allowing the model to remain abstract and decoupled from specific implementations in accordance with [RFC3444]. This abstraction supports different DM implementations, such as YANG or IPFIX [RFC7011], while ensuring consistency across implementations. Using YANG for the IM enables this abstraction, leverages the community's familiarity with its syntax, and ensures lossless translation to the corresponding YANG data model, which is defined in Section 5.

> Design note: In order to ease reuse of the IM structure by DMs but without requiring that these DMs to parse the "sx" structure defined in [RFC8791], main reusable nodes are defined in a common module (Section 4.3) while the main IM structure is defined in Section 4.4.

4.1. Structure

The IM defines a hierarchical classification scheme for packet discards, which captures where in a device the discards are accounted (component), in which direction of traffic they were flowing (direction), whether they were successfully processed or discarded (type), what protocol layer they belong to (layer), and the specific reason for any discards (subtypes). This structure enables both high-level monitoring of total discards (i.e., aggregates) and more detailed triage to map to mitigation actions.

The abstract structure of the IM is depicted in Figure 1. The full YANG tree diagram of the IM is provided in Appendix C.

```
module: ietf-packet-discard-reporting-sx

  structure packet-discard-reporting:
    +-- control-plane {pdr-common:control-plane-stats}?
    |  +-- traffic* [direction]
    |  |  ...
    |  +-- discards* [direction]
    |     ...
    +-- interface* [name] {pdr-common:interface-stats}?
    |  +-- name        string
    |  +-- traffic* [direction]
    |  |  +-- direction    identityref
    |  |  +-- l2
    |  |  |  ...
    |  |  +-- l3
    |  |  |  ...
```

```
|  |  +-- qos
|  |     ...
|  +-- discards* [direction]
|     +-- direction    identityref
|     +-- l2
|     |  ...
|     +-- l3
|     |  ...
|     +-- errors
|     |  +-- l2
|     |  |  ...
|     |  +-- l3
|     |  |  ...
|     |  +-- internal
|     |     ...
|     +-- policy
|     |  +-- l2
|     |  |  ...
|     |  +-- l3
|     |     ...
|     +-- no-buffer
|           ...
+-- flow* [direction] {pdr-common:flow-reporting}?
|  +-- direction    identityref
|  +-- traffic
|  |  +-- l2
|  |  |  ...
|  |  +-- l3
|  |  |  ...
|  |  +-- qos
|  |     ...
|  +-- discards
|     +-- l2
|     |  ...
|     +-- l3
|     |  ...
|     +-- errors
|     |  +-- l2
|     |  |  ...
|     |  +-- l3
|     |  |  ...
|     |  +-- internal
|     |     ...
|     +-- policy
|     |  +-- l2
|     |  |  ...
|     |  +-- l3
|     |     ...
```

```
|     +-- no-buffer
|           ...
+-- device {pdr-common:device-stats}?
   +-- traffic
   |  +-- l2
   |  |  ...
   |  +-- l3
   |  |  ...
   |  +-- qos
   |     ...
   +-- discards
      +-- l2
      |  ...
      +-- l3
      |  ...
      +-- errors
      |  +-- l2
      |  |  ...
      |  +-- l3
      |  |  ...
      |  +-- internal
      |     ...
      +-- policy
      |  +-- l2
      |  |  ...
      |  +-- l3
      |     ...
      +-- no-buffer
            ...
```

Figure 1: Abstract IM Tree Structure

The discard reporting can be organized into several types: control plane, interface, flow, and device. In order to allow for better mapping to underlying DMs, the IM supports a set of "features" to control the supported type.

A complete classification path follows the pattern: component/direction/type/layer/subtype/sub-subtype/.../metric. Appendix A illustrates where these discards typically occur in a network device. The elements of the tree are defined as follows:

* Component:

  - control-plane: discards of traffic to or from a device’s control plane.

- interface: discards of traffic to or from a specific network interface.
- flow: discards of traffic associated with a specific traffic flow.
- device: discards of traffic transiting the device.

* Direction:
  - ingress: counters for incoming packets or frames.
  - egress: counters for outgoing packets or frames.
* Type:
  - traffic: counters for successfully received or transmitted packets or frames.
  - discards: counters for packets or frames that were dropped.
* Layer:
  - l2: Layer 2 traffic and discards. This covers both frame and byte counts.
  - l3: Layer 3 traffic and discards. This covers both packet and byte counts.

The hierarchical structure allows for future extensions while maintaining backward compatibility. New discard types can be added as new branches without affecting existing implementations.

The corresponding YANG module is defined in Section 4.4.

4.2. Subtype Definitions

discards/policy/: These are intended discards, meaning packets dropped due to a configured policy, including: ACLs, traffic policers, unicast Reverse Path Forwarding (uRPF) checks, Distributed Denial-of-Service (DDoS) protection rules, and explicit null routes. In practice, ingress DDoS protection policies are often realized using mechanisms such as ingress filtering and uRPF ([RFC2827], [RFC3704], and [RFC8704]), remotely triggered blackholing ([RFC3882], [RFC5635]), or BGP Flow Specification窶濺ased filters ([RFC8955], [RFC8956], and [RFC9117]); all such policy-driven discards are reported under this class.

discards/errors/: These are unintended discards due to errors in processing packets or frames. There are multiple sub-classes:

* discards/errors/l2/rx/: These are frames discarded due to errors in the received Layer 2 frame, including: Cyclic Redundancy Check (CRC) errors, invalid Media Access Control (MAC) addresses, invalid VLAN tags, frame size violations and other malformed frame conditions.

* discards/errors/l3/rx/: These discards occur due to errors in the received packet, indicating an upstream problem rather than an issue with the device dropping the errored packets, including: header checksum errors, MTU exceeded, invalid packet errors (i.e., incorrect version, incorrect header length, invalid options, and other malformed packet conditions).

* discards/errors/l3/ttl-expired: These discards occur due to TTL (or Hop limit) expiry. These can occur, e.g., for the following reasons: normal trace-route operations, end-system TTL/Hop limit set too low, or routing loops in the network.

* discards/errors/l3/no-route/: These discards occur due to a packet not matching any route in the routing table, e.g., which may be due to routing configuration errors or may be transient discards during convergence.

* discards/errors/internal/: These discards occur due to internal device issues, including: parity errors in device memory or other internal hardware errors. Any errored discards not explicitly assigned to other classes are also accounted for here.

discards/no-buffer/: These are discards due to buffer exhaustion (that is congestion related discards). These can be tail-drop discards or due to an active queue management algorithm, such as Random Early Detection (RED) [RED93] or Controlled Delay (CoDel) [RFC8289].

An example of possible signal-to-mitigation action mapping is provided in Appendix B.

4.3. "ietf-packet-discard-reporting-common" YANG Module

The "ietf-packet-discard-reporting-common" module imports "ietf-yang-types" defined in [RFC9911].

```
<CODE BEGINS>
  file "ietf-packet-discard-reporting-common@2026-03-03.yang"
module ietf-packet-discard-reporting-common {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:"
          + "ietf-packet-discard-reporting-common";
  prefix pdr-common;

  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 9911: Common YANG Data Types";
  }

  organization
    "IETF OPSAWG (Operations and Management Area Working Group)";
  contact
    "WG Web:   https://datatracker.ietf.org/wg/opsawg/
     WG List:  OPSAWG <mailto:opsawg@ietf.org>

     Editor:   John Evans
               <mailto:jevanamz@amazon.co.uk>

     Editor:   Oleksandr Pylypenko
               <mailto:opylypenko@nvidia.com>

     Author:   Jeffrey Haas
               <mailto:jhaas@juniper.net>

     Author:   Aviran Kadosh
               <mailto:akadosh@cisco.com>

     Editor:   Mohamed Boucadair
               <mailto:mohamed.boucadair@orange.com>";
  description
    "This module defines a common YANG module for packet discard
     reporting.

     Copyright (c) 2026 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.

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

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

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

      revision 2026-03-03 {
        description
          "Initial revision.";
        reference
          "RFC XXXX: Information and Data Models for Packet Discard
                     Reporting";
      }

      /*
       * Features
       */

      feature control-plane-stats {
        description
          "Indicates support of control plane discard statistics.";
      }

      feature interface-stats {
        description
          "Indicates support of interface discard statistics.";
      }

      feature flow-reporting {
        description
          "Indicates support of flow discard reporting.";
      }

      feature device-stats {
        description
          "Indicates support of global device discard statistics.";
      }

      /*
       * Identities
       */

      identity direction {
        description
          "Defines a direction for the reported statistics.";
      }
```

```
identity ingress {
  base direction;
  description
    "Reports statistics for the received packets from
     the network.";
}

identity egress {
  base direction;
  description
    "Reports statistics for the sent packets to
     to the network.";
}

identity address-family {
  description
    "Defines a type for the address family.

     This identity is defined here rather than importing
     it from other YANG modules to simplify implementations
     and avoid inheriting dependencies of those modules.

     Additional address families can be added by defining
     identities derived from this base identity, without
     affecting existing implementations.";
}

identity ip {
  base address-family;
  description
    "Identity for IP address family.";
}

identity ipv4 {
  base ip;
  description
    "Identity for IPv4 address family.";
}

identity ipv6 {
  base ip;
  description
    "Identity for IPv6 address family.";
}

/*
 * Groupings
 */
```

```
  grouping basic-packets {
    description
      "Grouping for packet counters.";
    leaf packets {
      type yang:counter64;
      description
        "Number of packets.";
    }
  }

  grouping basic-packets-bytes {
    description
      "Grouping for packet and byte counters.";
    uses basic-packets;
    leaf bytes {
      type yang:counter64;
      description
        "Number of bytes.";
    }
  }

  grouping basic-frames {
    description
      "Grouping for Layer 2 frame counters.";
    leaf frames {
      type yang:counter64;
      description
        "Number of Layer 2 frames.";
    }
  }

  grouping l2-traffic {
    description
      "Grouping for Layer 2 frame and byte counters.";
    uses basic-frames;
    leaf bytes {
      type yang:counter64;
      description
        "Number of Layer 2 bytes.";
    }
  }

  grouping l3-traffic {
    description
      "Layer 3 traffic counters per address family.";
    list address-family-stat {
      key "address-family";
      description
```

```
          "Reports per address family traffic counters.";
        leaf address-family {
          type identityref {
            base address-family;
          }
          description
            "Specifies an address family.";
        }
        uses basic-packets-bytes;
        container unicast {
          description
            "Unicast traffic counters.";
          uses basic-packets-bytes;
        }
        container multicast {
          description
            "Multicast traffic counters.";
          uses basic-packets-bytes;
        }
        container broadcast {
          when "derived-from-or-self(../address-family, "
             + "'pdr-common:ipv4')" {
            description
              "Only applicable for IPv4.";
          }
          description
            "Broadcast traffic counters.";
          uses basic-packets-bytes;
        }
      }
    }

    grouping class-list {
      description
        "Class-based traffic counters.";
      list class {
        key "id";
        min-elements 1;
        description
          "Class traffic counters.";
        leaf id {
          type string;
          description
            "Indicates a Quality of Service (QoS) class
             identifier.";
        }
        uses basic-packets-bytes;
      }
```

```
  }

  grouping qos {
    description
      "QoS traffic counters.";
    container qos {
      presence "QoS statistics are available.";
      description
        "Per-class QoS traffic counters.";
      uses class-list;
    }
  }

  grouping traffic {
    description
      "All traffic counters.";
    container l2 {
      description
        "Layer 2 traffic counters.";
      uses l2-traffic;
    }
    container l3 {
      description
        "Layer 3 traffic counters.";
      uses l3-traffic;
    }
    uses qos;
  }

  grouping errors-l2-rx {
    description
      "Layer 2 ingress frame error discard counters.";
    container rx {
      description
        "Layer 2 ingress frame receive error discard
         counters.";
      leaf frames {
        type yang:counter64;
        description
          "The number of frames discarded due to errors
           with the received frame.";
      }
      leaf crc-error {
        type yang:counter64;
        description
          "The number of received frames discarded due to
            Cyclic Redundancy Check (CRC) error.";
      }
```

```
        leaf invalid-mac {
          type yang:counter64;
          description
            "The number of received frames discarded due to
             an invalid Media Access Control (MAC) address.";
        }
        leaf invalid-vlan {
          type yang:counter64;
          description
            "The number of received frames discarded due to
             an invalid VLAN tag.";
        }
        leaf invalid-frame {
          type yang:counter64;
          description
            "The number of invalid received frames discarded due to
             other reasons, not limited to: malformed frames,
             frame-size violations.";
        }
      }
    }

    grouping errors-l3-rx {
      description
        "Layer 3 ingress packet error discard counters.";
      container rx {
        description
          "Layer 3 ingress packet receive error discard
           counters.";
        leaf packets {
          type yang:counter64;
          description
            "The number of Layer 3 packets discarded due to
             errors in the received packet.";
        }
        leaf checksum-error {
          type yang:counter64;
          description
            "The number of received packets discarded due
             to a checksum error.";
        }
        leaf mtu-exceeded {
          type yang:counter64;
          description
            "The number of received packets discarded due to
             MTU exceeded.";
        }
        leaf invalid-packet {
```

```
          type yang:counter64;
          description
            "The number of received invalid packets discarded due
             to other reasons, not limited to: invalid packet length,
             invalid header fields, invalid options, invalid protocol
             version, invalid flags or control bits, malformed
             packets.";
        }
      }
      leaf ttl-expired {
        type yang:counter64;
        description
          "The number of received packets discarded due to
           expired TTL or Hop Limit exceeded.";
      }
      leaf no-route {
        type yang:counter64;
        description
          "The number of received packets discarded due to not
           matching a valid route.";
      }
      leaf invalid-sid {
        type yang:counter64;
        description
          "The number of received packets discarded due to an
           invalid Segment Routing over IPv6 (SRv6) segment
           identifier (SID).
           For SR-MPLS, invalid SIDs have to be accounted
           under invalid-label.";
      }
      leaf invalid-label {
        type yang:counter64;
        description
          "The number of received packets discarded due to an
           invalid MPLS label.";
      }
    }

    grouping errors-l3-int {
      description
        "Internal error discard counters.";
      leaf packets {
        type yang:counter64;
        description
          "The number of packets discarded due to internal
           errors.";
      }
      leaf parity-error {
```

```
        type yang:counter64;
        description
          "The number of packets discarded due to parity
           errors.";
      }
    }

    grouping errors-l2-tx {
      description
        "Layer 2 transmit error discard counters.";
      container tx {
        description
          "Layer 2 transmit frame error discard counters.";
        leaf frames {
          type yang:counter64;
          description
            "The number of Layer 2 frames discarded due to
             errors when transmitting.";
        }
      }
    }

    grouping errors-l3-tx {
      description
        "Layer 3 transmit error discard counters.";
      container tx {
        description
          "Layer 3 transmit packet error discard counters.";
        leaf packets {
          type yang:counter64;
          description
            "The number of Layer 3 packets discarded due to
             errors when transmitting.";
        }
      }
    }

    grouping errors {
      description
        "Error discard counters.";
      container l2 {
        description
          "Layer 2 frame error discard counters.";
        uses errors-l2-rx;
        uses errors-l2-tx;
      }
      container l3 {
        description
```

```
          "Layer 3 packet error discard counters.";
        uses errors-l3-rx;
        uses errors-l3-tx;
      }
      container internal {
        description
          "Internal error discard counters.";
        uses errors-l3-int;
      }
    }

    grouping policy-l2 {
      description
        "Layer 2 policy frame discard counters.";
      leaf frames {
        type yang:counter64;
        description
          "The number of Layer 2 frames discarded due
           to policy.";
      }
      leaf acl {
        type yang:counter64;
        description
          "The number of frames discarded due to Layer 2
           Access Control Lists (ACLs).";
      }
    }

    grouping policy-l3 {
      description
        "Layer 3 policy packet discard counters.";
      leaf packets {
        type yang:counter64;
        description
          "The number of Layer 3 packets discarded due to policy.";
      }
      leaf acl {
        type yang:counter64;
        description
          "The number of packets discarded due to Layer 3 ACLs.";
      }
      container policer {
        description
          "The number of packets discarded due to policer
           violations.";
        uses basic-packets-bytes;
        container classes {
          presence "Per-class policer statistics are available.";
```

```
          description
            "Per-class policer discard counters.";
          uses class-list;
        }
      }
      leaf null-route {
        type yang:counter64;
        description
          "The number of packets discarded due to matching
           a null route.";
      }
      leaf rpf {
        type yang:counter64;
        description
          "The number of packets discarded due to failing
           Reverse Path Forwarding (RPF) check.";
      }
      leaf ddos {
        type yang:counter64;
        description
          "The number of packets discarded due to Distributed
           Denial-of-Service (DDoS) protection policies.";
      }
    }

    grouping discards {
      description
        "Discard counters.";
      container l2 {
        description
          "Layer 2 frame discard counters.";
        uses l2-traffic;
      }
      container l3 {
        description
          "Layer 3 packet discard counters.";
        uses l3-traffic;
      }
      container errors {
        description
          "Error discard counters.";
        uses errors;
      }
      container policy {
        description
          "Policy-related discard counters.";
        uses policy;
      }
```

```
      container no-buffer {
        description
          "Discard counters due to buffer unavailability.";
        uses qos;
      }
    }

    grouping policy {
      description
        "Policy-related discard counters.";
      container l2 {
        description
          "Layer 2 policy frame discard counters.";
        uses policy-l2;
      }
      container l3 {
        description
          "Layer 3 policy packet discard counters.";
        uses policy-l3;
      }
    }

    grouping traffic-and-discards {
      description
        "Specifies overall traffic and discard counters.";
      container traffic {
        description
          "Traffic counters.";
        uses traffic;
      }
      container discards {
        description
          "Discard counters.";
        uses discards;
      }
    }

    grouping interface {
      description
        "Interface-level traffic and discard counters.";
      list traffic {
        key "direction";
        description
          "Traffic counters.";
        leaf direction {
          type identityref {
            base direction;
          }
```

```
          description
            "Specifies a direction.";
        }
        uses traffic;
      }
      list discards {
        key "direction";
        description
          "Discard counters.";
        leaf direction {
          type identityref {
            base direction;
          }
          description
            "Specifies a direction.";
        }
        uses discards;
      }
    }

    grouping control-plane {
      description
        "Control plane packet counters.";
      list traffic {
        key "direction";
        description
          "Total control plane packets.";
        leaf direction {
          type identityref {
            base direction;
          }
          description
            "Specifies a direction.";
        }
        uses basic-packets-bytes;
      }
      list discards {
        key "direction";
        description
          "Control plane packet discard counters.";
        leaf direction {
          type identityref {
            base direction;
          }
          description
            "Specifies a direction.";
        }
        uses basic-packets-bytes;
```

```
        container policy {
          description
            "Number of control plane packets discarded due to policy.";
          uses basic-packets;
        }
      }
    }
  }
  <CODE ENDS>
```

4.4. "ietf-packet-discard-reporting-sx" YANG Module

The "ietf-packet-discard-reporting-sx" module uses the "sx" structure defined in [RFC8791] and also imports the "ietf-packet-discard-reporting-common" module (Section 4.3).

```
  <CODE BEGINS> file "ietf-packet-discard-reporting-sx@2026-03-03.yang"
  module ietf-packet-discard-reporting-sx {
    yang-version 1.1;
    namespace
      "urn:ietf:params:xml:ns:yang:ietf-packet-discard-reporting-sx";
    prefix pdr-sx;

    import ietf-packet-discard-reporting-common {
      prefix pdr-common;
      reference
        "RFC XXXX: Information and Data Models for Packet Discard
                   Reporting";
    }
    import ietf-yang-structure-ext {
      prefix sx;
      reference
        "RFC 8791: YANG Data Structure Extensions";
    }

    organization
      "IETF OPSAWG (Operations and Management Area Working Group)";
    contact
      "WG Web:   https://datatracker.ietf.org/wg/opsawg/
       WG List:  OPSAWG <mailto:opsawg@ietf.org>

       Editor:   John Evans
                 <mailto:jevanamz@amazon.co.uk>

       Editor:   Oleksandr Pylypenko
                 <mailto:opylypenko@nvidia.com>

       Author:   Jeffrey Haas
```

```
                <mailto:jhaas@juniper.net>

      Author:   Aviran Kadosh
                <mailto:akadosh@cisco.com>

      Editor:   Mohamed Boucadair
                <mailto:mohamed.boucadair@orange.com>";
    description
      "This module defines an information model for packet discard
       reporting.

       Copyright (c) 2026 IETF Trust and the persons identified as
       authors of the code.  All rights reserved.

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

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

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

    revision 2026-03-03 {
      description
        "Initial revision.";
      reference
        "RFC XXXX: Information and Data Models for Packet Discard
                   Reporting";
    }

    /*
     * Main structure definition
     */

    sx:structure packet-discard-reporting {
      description
        "Specifies the abstract structure of packet discard
         reporting data.";
      container control-plane {
        if-feature "pdr-common:control-plane-stats";
        description
          "Control plane packet counters.";
```

```
        uses pdr-common:control-plane;
      }
      list interface {
        if-feature "pdr-common:interface-stats";
        key "name";
        description
          "Indicates a list of interfaces for which packet
           discard reporting data is provided.";
        leaf name {
          type string;
          description
            "Indicates the name of the interface.";
        }
        uses pdr-common:interface;
      }
      list flow {
        if-feature "pdr-common:flow-reporting";
        key "direction";
        leaf direction {
          type identityref {
            base pdr-common:direction;
          }
          description
            "Specifies a direction.";
        }
        description
          "Flow packet counters.";
        uses pdr-common:traffic-and-discards;
      }
      container device {
        if-feature "pdr-common:device-stats";
        description
          "Device level packet counters.";
        uses pdr-common:traffic-and-discards;
      }
    }
  }
  <CODE ENDS>
```

5. Data Model (DM)

This DM implements the IM defined in Section 4 for the interface, device, and control-plane components. It is a device model per Section 2.1 of [RFC8969]. Specifically, it is a device-local (network element) operational state model: counters are scoped to a single device (interfaces and control plane).

The IM defines the abstract classification tree using YANG data structure extensions [RFC8791]. This DM imports that module and reuses the same groupings and hierarchy of components, directions, layers, and discard classes, attaching them via augment statements to existing YANG modules for routing, interfaces, and logical network elements. The flow component is defined only in the IM for use by flow-oriented data models and are not instantiated in this DM.

5.1. Structure

There is a direct mapping between the IM components and their DM implementations, with each component in the hierarchy represented by corresponding YANG containers and leaf data nodes. The abstract tree is shown in Figure 2.

```
module: ietf-packet-discard-reporting

  augment /rt:routing/rt:control-plane-protocols
            /rt:control-plane-protocol:
    +--ro discard-stats {control-plane-stats}?
       +--ro discard-order-capability*   identityref
       +--ro traffic* [direction]
       |  ...
       +--ro discards* [direction]
          ...
  augment /if:interfaces/if:interface/if:statistics:
    +--ro discard-order-capability*   identityref {interface-stats}?
    +--ro traffic* [direction] {interface-stats}?
    |  +--ro direction    identityref
    |  +--ro l2
    |  |  ...
    |  +--ro l3
    |  |  ...
    |  +--ro qos!
    |     +--ro class* [id]
    |        ...
    +--ro discards* [direction] {interface-stats}?
       +--ro direction    identityref
       +--ro l2
       |  ...
       +--ro l3
       |  ...
       +--ro errors
       |  +--ro l2
       |  |  ...
       |  +--ro l3
       |  |  ...
       |  +--ro internal
```

```
          |     ...
          +--ro policy
          |  +--ro l2
          |  |  ...
          |  +--ro l3
          |     ...
          +--ro no-buffer
             +--ro qos!
                +--ro class* [id]
                   ...
     augment /lne:logical-network-elements/lne:logical-network-element:
       +--ro discard-stats {device-stats}?
          +--ro discard-order-capability*   identityref
          +--ro traffic
          |  +--ro l2
          |  |  ...
          |  +--ro l3
          |  |  ...
          |  +--ro qos!
          |     +--ro class* [id]
          |        ...
          +--ro discards
             +--ro l2
             |  ...
             +--ro l3
             |  ...
             +--ro errors
             |  +--ro l2
             |  |  ...
             |  +--ro l3
             |  |  ...
             |  +--ro internal
             |     ...
             +--ro policy
             |  +--ro l2
             |  |  ...
             |  +--ro l3
             |     ...
             +--ro no-buffer
                +--ro qos!
                   +--ro class* [id]
                      ...
```

Figure 2: Abstract DM Tree Structure

The full tree structure is provided in Appendix D.

5.2. Implementation Requirements

The following requirements apply to the implementation of the DM and are intended to ensure consistent implementation across different vendors and platforms while allowing for platform-specific optimisations where needed. While the DM defines a comprehensive set of counters and statistics, implementations MAY support a subset of the defined features based on device capabilities and operational requirements. However, implementations MUST clearly document which features are supported and how they map to the DM.

Requirements 1-13 relate to packets forwarded or discarded by the device, while requirement 14 relates to packets destined for or originating from the device:

1. All instances of Layer 2 frame or Layer 3 packet receipt, transmission, and discards MUST be accounted for.

2. All instances of Layer 2 frame or Layer 3 packet receipt, transmission, and discards SHOULD be attributed to the physical or logical interface of the device where they occur. Where they cannot be attributed to the interface, they MUST be attributed to the device.

3. An individual frame MUST only be accounted for by either the Layer 2 traffic class or the Layer 2 discard classes within a single direction or context, i.e., ingress or egress or device. This is to avoid double counting.

4. An individual packet MUST only be accounted for by either the Layer 3 traffic class or the Layer 3 discard classes within a single direction or context, i.e., ingress or egress or device. This is to avoid double counting.

5. A frame accounted for at Layer 2 MUST NOT be accounted for at Layer 3 and vice versa. This is to avoid double counting.

6. The aggregate Layer 2 and Layer 3 traffic and discard classes SHOULD account for all underlying frames or packets received, transmitted, and discarded across all other classes. There might be exceptions when distinct discontinuity times are observed for more granular discards.

7. The aggregate QoS traffic and no-buffer discard classes MUST account for all underlying packets received, transmitted, and discarded across all other classes.

8. In addition to the Layer 2 and Layer 3 aggregate classes, an individual discarded packet MUST only account against a single error, policy, or no-buffer discard subclass.

9. When there are multiple reasons for discarding a packet, the ordering of discard class reporting MUST be defined. Typically, this can be exposed by an implementation by means of discard-order-capability.

10. If Diffserv [RFC2475] is not used, no-buffer discards MUST be reported as class[id="0"], which represents the default class.

11. When traffic is mirrored, the discard metrics MUST account for the original traffic rather than the reflected traffic.

12. No-buffer discards can be realized differently with different memory architectures. Whether a no-buffer discard is attributed to ingress or egress can differ accordingly. For successful auto-mitigation, discards due to an egress interface congestion MUST be reportable on egress, while discards due to device-level congestion (e.g., due to exceeding the device forwarding rate) MUST be reportable on ingress.

13. When the ingress and egress headers differ (for example, at a tunnel endpoint), the discard class attribution MUST relate to the outer header at the point of discard.

14. Traffic to the device control plane (to-CPU) has its own class. Traffic from the device control plane (from-CPU) is accounted for by origin, independent of the forwarding mechanism (e.g., any egress policer it traverses), and MUST also be accounted for in the same way as other egress traffic.

5.3. Usage Examples

This section assumes that no class of service is implemented.

If all of the requirements listed in Section 5.2 are met, a "good" unicast IPv4 packet received would increment:

* interface/traffic[direction="ingress"]/l3/address-family-stat[address-family="ipv4"]/unicast/packets

* interface/traffic[direction="ingress"]/l3/address-family-stat[address-family="ipv4"]/unicast/bytes

* interface/traffic[direction="ingress"]/qos/class[id="0"]/packets

* interface/traffic[direction="ingress"]/qos/class[id="0"]/bytes

A received unicast IPv6 packet discarded due to Hop Limit expiry would increment:

* interface/traffic[direction="ingress"]/l3/address-family-stat[address-family="ipv6"]/unicast/packets

* interface/traffic[direction="ingress"]/l3/address-family-stat[address-family="ipv6"]/unicast/bytes

* interface/discards[direction="ingress"]/errors/l3/ttl-expired

An IPv4 packet discarded on egress due to no buffers would increment:

* interface/discards[direction="egress"]/l3/address-family-stat[address-family="ipv4"]/unicast/packets

* interface/discards[direction="egress"]/l3/address-family-stat[address-family="ipv4"]/unicast/bytes

* interface/discards[direction="egress"]/no-buffer/class[id="0"]/packets

* interface/discards[direction="egress"]/no-buffer/class[id="0"]/bytes

A multicast IPv6 packet dropped due to RPF check failure would increment:

* interface/discards[direction="ingress"]/l3/address-family-stat[address-family="ipv6"]/multicast/packets

* interface/discards[direction="ingress"]/l3/address-family-stat[address-family="ipv6"]/multicast/bytes

* interface/discards[direction="ingress"]/policy/l3/rpf

A "good" Layer-2 frame received would increment:

* interface/traffic[direction="ingress"]/l2/frames

* interface/traffic[direction="ingress"]/l2/bytes

* interface/traffic[direction="ingress"]/qos/class[id="0"]/packets

* interface/traffic[direction="ingress"]/qos/class[id="0"]/bytes

5.4. "ietf-packet-discard-reporting" YANG Module

The "ietf-packet-discard-reporting" module imports "ietf-packet-discard-reporting-common" (Section 4.3), "ietf-netconf-acm" [RFC8341], "ietf-interfaces" [RFC8343], "ietf-routing" [RFC8349], and "ietf-logical-network-element" [RFC8530].

<CODE BEGINS> file "ietf-packet-discard-reporting@2026-03-03.yang"

```
module ietf-packet-discard-reporting {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-packet-discard-reporting";
  prefix pdr;

  import ietf-packet-discard-reporting-common {
    prefix pdr-common;
    reference
      "RFC XXXX: Information and Data Models for Packet Discard
                 Reporting";
  }
  import ietf-netconf-acm {
    prefix nacm;
    reference
      "RFC 8341: Network Configuration Access Control Model";
  }
  import ietf-interfaces {
    prefix if;
    reference
      "RFC 8343: A YANG Data Model for Interface Management";
  }
  import ietf-routing {
    prefix rt;
    reference
      "RFC 8349: A YANG Data Model for Routing Management
                 (NMDA Version)";
  }
  import ietf-logical-network-element {
    prefix lne;
    reference
      "RFC 8530: YANG Model for Logical Network Elements";
  }

  organization
    "IETF OPSAWG (Operations and Management Area Working Group)";
  contact
    "WG Web:   https://datatracker.ietf.org/wg/opsawg/
     WG List:  OPSAWG <mailto:opsawg@ietf.org>
```

```
        Editor:   John Evans
                  <mailto:jevanamz@amazon.co.uk>

        Editor:   Oleksandr Pylypenko
                  <mailto:opylypenko@nvidia.com>

        Author:   Jeffrey Haas
                  <mailto:jhaas@juniper.net>

        Author:   Aviran Kadosh
                  <mailto:akadosh@cisco.com>

        Editor:   Mohamed Boucadair
                  <mailto:mohamed.boucadair@orange.com>";
    description
      "This module defines a data model for packet discard reporting.

       Copyright (c) 2026 IETF Trust and the persons identified as
       authors of the code.  All rights reserved.

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

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

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

    revision 2026-03-03 {
      description
        "Initial revision.";
      reference
        "RFC XXXX: Information and Data Models for Packet Discard
                   Reporting";
    }

    /*
     * Identities
     */

    identity discard-class {
      description
```

```
      "Base identity to identify the discard class.";
  }

  identity layer2 {
    base discard-class;
    description
      "Indicates a Layer 2 discard.";
  }

  identity layer3 {
    base discard-class;
    description
      "Indicates a Layer 3 discard.";
  }

  identity internal {
    base discard-class;
    description
      "Indicates an internal discard.";
  }

  identity policy {
    base discard-class;
    description
      "Indicates a discard due to a policy.";
  }

  identity no-buffer {
    base discard-class;
    description
      "Indicates a discard due to buffer unavailability.";
  }

  /*
   * Groupings
   */

  grouping discard-order-policy {
    description
      "Defines the implementation-specific precedence of discard
       classes when multiple discard reasons apply to a single
       packet.

       The list is ordered from highest to lowest precedence.";

    leaf-list discard-order-capability {
      type identityref {
        base discard-class;
```

```
        }
        config false;
        description
          "The discard class identity that has this precedence.";
      }
    }

    /*
     * Main structure definition
     */

    augment "/rt:routing/rt:control-plane-protocols"
          + "/rt:control-plane-protocol" {
      if-feature "pdr-common:control-plane-stats";
      description
        "Adds control plane discard counters.";
      container discard-stats {
        nacm:default-deny-all;
        config false;
        description
          "List of control plane discard counters.";
        uses discard-order-policy;
        uses pdr-common:control-plane;
      }
    }
    augment "/if:interfaces/if:interface/if:statistics" {
      if-feature "pdr-common:interface-stats";
      description
        "Adds packet discard reporting to the interface statistics.";
      uses discard-order-policy;
      uses pdr-common:interface;
    }
    augment "/lne:logical-network-elements"
          + "/lne:logical-network-element" {
      if-feature "pdr-common:device-stats";
      description
        "Adds device level packet counters.";

      container discard-stats {
        nacm:default-deny-all;
        config false;
        description
          "List of device level discard counters.";
        uses discard-order-policy;
        uses pdr-common:traffic-and-discards;
      }
    }
  }
```

<CODE ENDS>

6. Operational Considerations

6.1. Deployment Experience

This section captures practical insights gained from implementing the model across multiple vendors' platforms, as guidance for future implementers and operators:

1. The number and granularity of discard classes defined in the IM represent a compromise. It aims to provide sufficient detail to enable appropriate automated actions while avoiding excessive detail, which may hinder quick problem identification. Additionally, it helps to limit the quantity of data produced per interface, constraining the data volume and device CPU impacts. While further granularity is possible, the defined schema has generally proven to be sufficient for the task of mitigating unintended packet loss.

2. There are many possible ways to define the discard classification tree. For example, an approach is to use a multi-rooted tree, rooted in each protocol. Instead, a better approach is to define a tree where protocol discards and causal discard classes are accounted for orthogonally. This decision reduces the number of combinations of classes and has proven sufficient for determining mitigation actions.

3. Platforms often account for the number of packets discarded where the TTL has expired (or IPv6 Hop Limit exceeded), and the device CPU has returned an ICMP Time Exceeded message [RFC4884]. There is typically a policer applied to limit the number of packets sent to the device CPU, however, which implicitly limits the rate of TTL discards that are processed. One method to account for all packet discards due to TTL expired, even those that are dropped by a policer when being forwarded to the CPU, is to use accounting of all ingress packets received with TTL=1 as a proxy measure.

4. Where no route discards are implemented with a default null route, separate discard accounting is required for any explicit null routes configured in order to differentiate between interface/ingress/discards/policy/null-route/packets and interface/ingress/discards/errors/no-route/packets.

5. It is useful to account separately for transit packets discarded by ACLs or policers, and packets discarded by ACLs or policers which limit the number of packets to the device control plane.

6. It is not possible to identify a configuration error (e.g., when intended discards are unintended) with device discard metrics alone. For example, additional context is needed to determine if ACL discards are intended or due to a misconfigured ACL (i.e., with configuration validation before deployment or by detecting a significant change in ACL discards after a configuration change compared to before).

7. Aggregate counters need to be able to deal with the possibility of discontinuities in the underlying counters.

8. While the classification tree is seven levels deep, a minimal implementation may only implement the top six.

6.2. Anchoring Flow Structure

The characterization of a flow depends on the underlying data model that adheres to the IM. From that standpoint, the IM does not make an assumption about flow characterization and identification. Future flow-oriented data models MUST ensure that the flow structure is anchored so that the discards are unambiguously associated with a flow.

7. Implementation Status

Note to RFC Editor: This section is to be removed before publication as an RFC.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in RFC 7942. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

7.1. Information Model Implementations

The IM defined in Section 4 has been implemented or mapped on at least nine hardware platforms across four vendors, including:

* Broadcom: Trident, Tomahawk 1, Tomahawk 3, Tomahawk 5

* Cisco: Q200L

* Juniper: MX, PTX, QFX

* Marvell: TL7

7.2. Data Model Implementations

A YANG-compliant open-source SLAX script implements a subset of the DM defined in Section 5 for Juniper MX routers. This implementation is available at:

* https://github.com/o-pylypenko-aws/draft-ietf-opsawg-discardmodel-sample/ (https://github.com/o-pylypenko-aws/draft-ietf-opsawg-discardmodel-sample/)

Practical observations from these implementations are reflected in Section 6.1.

8. Security Considerations

8.1. Information Model

The IM defined in Section 4.4 specifies a YANG module using [RFC8791] data extensions. As such, there are no additional security issues related to the YANG module that need to be considered.

The "ietf-packet-discard-reporting-common" YANG module defines a set of identities, types, and groupings. These nodes are intended to be reused by other YANG modules. The module by itself does not expose any data nodes that are writable, data nodes that contain read-only state, or RPCs. As such, there are no additional security issues related to the YANG module that need to be considered.

Modules that use the groupings that are defined in the "ietf-packet-discard-reporting-common" module should identify the corresponding security considerations.

8.2. Data Model

This section is modeled after the template described in Section 3.7.1 of [RFC9907].

The YANG module specified in Section 5.4 defines a data model that is designed to be accessed via YANG-based management protocols, such as Network Configuration Protocol (NETCONF) [RFC6241] and RESTCONF [RFC8040]. These YANG-based management protocols (1) have to use a secure transport layer (e.g., Secure Shell (SSH) [RFC4252], TLS [RFC8446], and QUIC [RFC9000]) and (2) have to use mutual authentication.

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

There are no particularly sensitive writable data nodes.

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

rt:control-plane-protocol/pdr:discard-stats, if:statistics/ pdr:traffic, if:statistics/pdr:discards, and lne:logical-network-element/pdr:discard-stats: Access to these data nodes would reveal information about the attacks to which an element is subject, misconfigurations, etc.

Also, an attacker who can inject packets can infer the efficiency of its attack by monitoring (the increase of) some discard counters (e.g., policy) and adjust its attack strategy accordingly.

9. IANA Considerations

IANA is requested to register the following URI in the "ns" subregistry within the "IETF XML Registry" [RFC3688]:

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

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

IANA is requested to register the following YANG module in the "YANG Module Names" subregistry [RFC6020] within the "YANG Parameters" registry:

Name: ietf-packet-discard-reporting-common
Namespace:
urn:ietf:params:xml:ns:yang:ietf-packet-discard-reporting-common
Prefix: pdr-common
Maintained by IANA? N
Reference: RFC XXXX

Name: ietf-packet-discard-reporting-sx
Namespace: urn:ietf:params:xml:ns:yang:ietf-packet-discard-reporting-sx
Prefix: pdr-sx
Maintained by IANA? N
Reference: RFC XXXX

Name: ietf-packet-discard-reporting
Namespace: urn:ietf:params:xml:ns:yang:ietf-packet-discard-reporting
Prefix: pdr
Maintained by IANA? N
Reference: RFC XXXX

10. References

10.1. Normative References

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

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, DOI 10.17487/RFC2475, December 1998, <https://www.rfc-editor.org/rfc/rfc2475>.

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

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

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/rfc/rfc7950>.

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

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/rfc/rfc8341>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/rfc/rfc8343>.

[RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, DOI 10.17487/RFC8349, March 2018, <https://www.rfc-editor.org/rfc/rfc8349>.

[RFC8530] Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X. Liu, "YANG Model for Logical Network Elements", RFC 8530, DOI 10.17487/RFC8530, March 2019, <https://www.rfc-editor.org/rfc/rfc8530>.

[RFC8791] Bierman, A., Bjテカrklund, M., and K. Watsen, "YANG Data Structure Extensions", RFC 8791, DOI 10.17487/RFC8791, June 2020, <https://www.rfc-editor.org/rfc/rfc8791>.

[RFC9911] Schテカnwテ、lder, J., Ed., "Common YANG Data Types", RFC 9911, DOI 10.17487/RFC9911, December 2025, <https://www.rfc-editor.org/rfc/rfc9911>.

10.2. Informative References

[RED93] Floyd, S. and V. Jacobson, "Random early detection gateways for congestion avoidance", August 1993, <https://ieeexplore.ieee.org/document/251892>.

[RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, <https://www.rfc-editor.org/rfc/rfc2827>.

[RFC2863] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, DOI 10.17487/RFC2863, June 2000, <https://www.rfc-editor.org/rfc/rfc2863>.

[RFC3444] Pras, A. and J. Schoenwaelder, "On the Difference between Information Models and Data Models", RFC 3444, DOI 10.17487/RFC3444, January 2003, <https://www.rfc-editor.org/rfc/rfc3444>.

[RFC3704] Baker, F. and P. Savola, "Ingress Filtering for Multihomed Networks", BCP 84, RFC 3704, DOI 10.17487/RFC3704, March 2004, <https://www.rfc-editor.org/rfc/rfc3704>.

[RFC3882] Turk, D., "Configuring BGP to Block Denial-of-Service Attacks", RFC 3882, DOI 10.17487/RFC3882, October 2004, <https://www.rfc-editor.org/rfc/rfc3882>.

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

[RFC4884] Bonica, R., Gan, D., Tappan, D., and C. Pignataro, "Extended ICMP to Support Multi-Part Messages", RFC 4884, DOI 10.17487/RFC4884, April 2007, <https://www.rfc-editor.org/rfc/rfc4884>.

[RFC5635] Kumari, W. and D. McPherson, "Remote Triggered Black Hole Filtering with Unicast Reverse Path Forwarding (uRPF)", RFC 5635, DOI 10.17487/RFC5635, August 2009, <https://www.rfc-editor.org/rfc/rfc5635>.

[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>.

[RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <https://www.rfc-editor.org/rfc/rfc7011>.

[RFC7270] Yourtchenko, A., Aitken, P., and B. Claise, "Cisco-Specific Information Elements Reused in IP Flow Information Export (IPFIX)", RFC 7270, DOI 10.17487/RFC7270, June 2014, <https://www.rfc-editor.org/rfc/rfc7270>.

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

[RFC8289] Nichols, K., Jacobson, V., McGregor, A., Ed., and J. Iyengar, Ed., "Controlled Delay Active Queue Management", RFC 8289, DOI 10.17487/RFC8289, January 2018, <https://www.rfc-editor.org/rfc/rfc8289>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/rfc/rfc8340>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

[RFC8704] Sriram, K., Montgomery, D., and J. Haas, "Enhanced Feasible-Path Unicast Reverse Path Forwarding", BCP 84, RFC 8704, DOI 10.17487/RFC8704, February 2020, <https://www.rfc-editor.org/rfc/rfc8704>.

[RFC8955] Loibl, C., Hares, S., Raszuk, R., McPherson, D., and M. Bacher, "Dissemination of Flow Specification Rules", RFC 8955, DOI 10.17487/RFC8955, December 2020, <https://www.rfc-editor.org/rfc/rfc8955>.

[RFC8956] Loibl, C., Ed., Raszuk, R., Ed., and S. Hares, Ed., "Dissemination of Flow Specification Rules for IPv6", RFC 8956, DOI 10.17487/RFC8956, December 2020, <https://www.rfc-editor.org/rfc/rfc8956>.

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

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

[RFC9117] Uttaro, J., Alcaide, J., Filsfils, C., Smith, D., and P. Mohapatra, "Revised Validation Procedure for BGP Flow Specifications", RFC 9117, DOI 10.17487/RFC9117, August 2021, <https://www.rfc-editor.org/rfc/rfc9117>.

[RFC9907] Bierman, A., Boucadair, M., Ed., and Q. Wu, "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 9907, DOI 10.17487/RFC9907, March 2026, <https://www.rfc-editor.org/rfc/rfc9907>.

Appendix A. Where Do Packets Get Dropped?

Understanding where packets are discarded in a network device is essential for interpreting discard signals and determining appropriate mitigation actions. Figure 3 depicts an example of where and why packets may be discarded in a typical single-ASIC, shared-buffered type device. While actual device architectures vary between vendors and platforms, with some using multiple ASICs, distributed forwarding, or different buffering architectures, this example illustrates the common processing stages where packets may be dropped. The logical model for classifying and reporting discards remains consistent regardless of the underlying hardware architecture.

Packets ingress on the left and egress on the right:

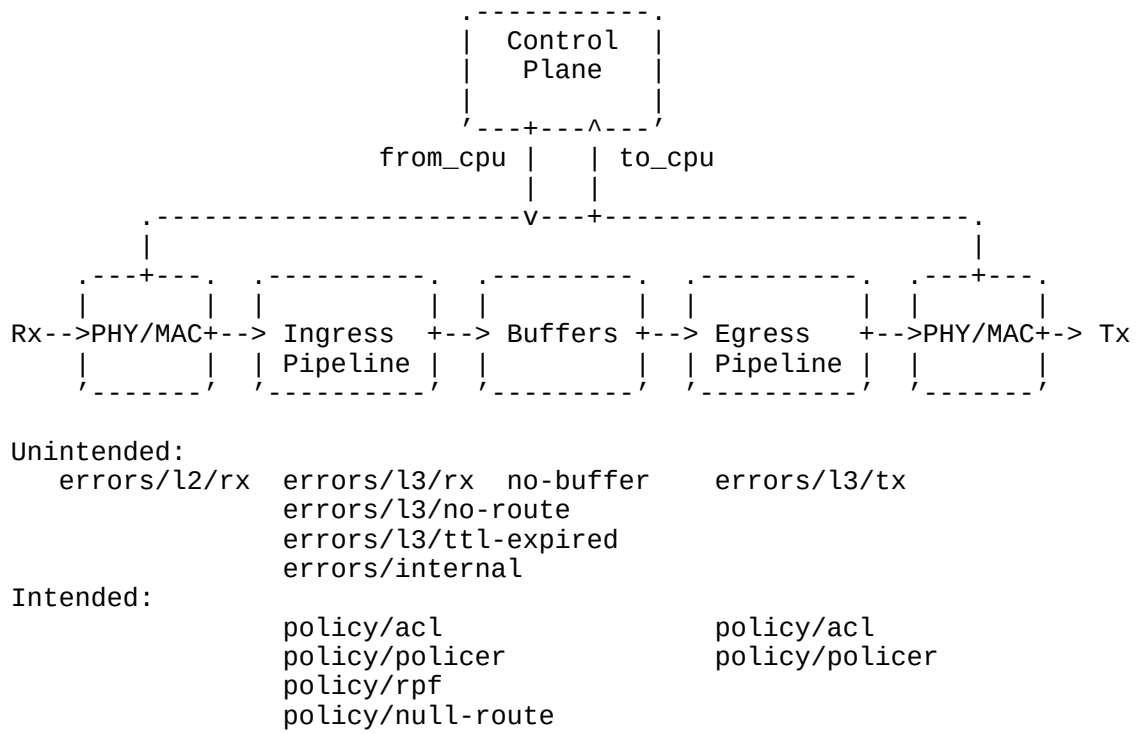

Figure 3: Example of Where Packets Get Dropped

See Appendix B for examples of how these discard signals map to root causes and mitigation actions.

Appendix B. Example Signal-to-mitigation Action Mapping

The effectiveness of automated mitigation depends on correctly mapping discard signals to root causes and appropriate actions. Tables 1 and 2 give example discard signal-to-mitigation action mappings based on the features described in Section 3.

DISCARD-CLASS	Discard cause	DISCARD-RATE	DISCARD-DURATION
ingress/discards/errors/l2/rx	Upstream device or link error	>Baseline	O(1min)
ingress/discards/errors/l3/ttl-expired	Tracert	<=Baseline	
ingress/discards/errors/l3/ttl-expired	Convergence	>Baseline	O(1s)
ingress/discards/errors/l3/ttl-expired	Routing loop	>Baseline	O(1min)
.*/policy/.*	Policy		
ingress/discards/errors/l3/no-route	Convergence	>Baseline	O(1s)
ingress/discards/errors/l3/no-route	Config error	>Baseline	O(1min)
ingress/discards/errors/l3/no-route	Invalid destination	>Baseline	O(10min)
ingress/discards/errors/internal	Device errors	>Baseline	O(1min)
egress/discards/no-buffer	Congestion	<=Baseline	
egress/discards/no-buffer	Congestion	>Baseline	O(1min)

Table 1: Example Signal-Cause-Mitigation Mapping (1)

DISCARD-CLASS	Unintended?	Possible actions
ingress/discards/errors/l2/ rx	Y	Take upstream link or device out-of-service
ingress/discards/errors/l3/ ttl-expired	N	no action
ingress/discards/errors/l3/ ttl-expired	Y	No action
ingress/discards/errors/l3/ ttl-expired	Y	Roll-back change
.*/policy/.*	N	No action
ingress/discards/errors/l3/ no-route	Y	No action
ingress/discards/errors/l3/ no-route	Y	Roll-back change
ingress/discards/errors/l3/ no-route	N	Escalate to operator
ingress/discards/errors/ internal	Y	Take device out-of-service
egress/discards/no-buffer	N	No action
egress/discards/no-buffer	Y	Bring capacity back into service or move traffic

Table 2: Example Signal-Cause-Mitigation Mapping (2)

The 'Baseline' in the 'DISCARD-RATE' column is both DISCARD-CLASS and network dependent.

Appendix C. Full Information Model Tree

The following YANG tree diagram shows the complete IM structure:

```
   module: ietf-packet-discard-reporting-sx

     structure packet-discard-reporting:
       +-- control-plane {pdr-common:control-plane-stats}?
       |  +-- traffic* [direction]
       |  |  +-- direction    identityref
       |  |  +-- packets?     yang:counter64
       |  |  +-- bytes?       yang:counter64
       |  +-- discards* [direction]
       |     +-- direction    identityref
       |     +-- packets?     yang:counter64
       |     +-- bytes?       yang:counter64
       |     +-- policy
       |        +-- packets?   yang:counter64
       +-- interface* [name] {pdr-common:interface-stats}?
       |  +-- name        string
       |  +-- traffic* [direction]
       |  |  +-- direction    identityref
       |  |  +-- l2
       |  |  |  +-- frames?   yang:counter64
       |  |  |  +-- bytes?    yang:counter64
       |  |  +-- l3
       |  |  |  +-- address-family-stat* [address-family]
       |  |  |     +-- address-family    identityref
       |  |  |     +-- packets?          yang:counter64
       |  |  |     +-- bytes?            yang:counter64
       |  |  |     +-- unicast
       |  |  |     |  +-- packets?   yang:counter64
       |  |  |     |  +-- bytes?     yang:counter64
       |  |  |     +-- multicast
       |  |  |     |  +-- packets?   yang:counter64
       |  |  |     |  +-- bytes?     yang:counter64
       |  |  |     +-- broadcast
       |  |  |        +-- packets?   yang:counter64
       |  |  |        +-- bytes?     yang:counter64
       |  |  +-- qos!
       |  |     +-- class* [id]
       |  |        +-- id         string
       |  |        +-- packets?   yang:counter64
       |  |        +-- bytes?     yang:counter64
       |  +-- discards* [direction]
       |     +-- direction    identityref
       |     +-- l2
       |     |  +-- frames?   yang:counter64
       |     |  +-- bytes?    yang:counter64
       |     +-- l3
       |     |  +-- address-family-stat* [address-family]
       |     |     +-- address-family    identityref
```

```
|     |     +-- packets?          yang:counter64
|     |     +-- bytes?            yang:counter64
|     |     +-- unicast
|     |     |  +-- packets?   yang:counter64
|     |     |  +-- bytes?     yang:counter64
|     |     +-- multicast
|     |     |  +-- packets?   yang:counter64
|     |     |  +-- bytes?     yang:counter64
|     |     +-- broadcast
|     |        +-- packets?   yang:counter64
|     |        +-- bytes?     yang:counter64
|     +-- errors
|     |  +-- l2
|     |  |  +-- rx
|     |  |  |  +-- frames?          yang:counter64
|     |  |  |  +-- crc-error?       yang:counter64
|     |  |  |  +-- invalid-mac?     yang:counter64
|     |  |  |  +-- invalid-vlan?    yang:counter64
|     |  |  |  +-- invalid-frame?   yang:counter64
|     |  |  +-- tx
|     |  |     +-- frames?   yang:counter64
|     |  +-- l3
|     |  |  +-- rx
|     |  |  |  +-- packets?          yang:counter64
|     |  |  |  +-- checksum-error?   yang:counter64
|     |  |  |  +-- mtu-exceeded?     yang:counter64
|     |  |  |  +-- invalid-packet?   yang:counter64
|     |  |  +-- ttl-expired?     yang:counter64
|     |  |  +-- no-route?        yang:counter64
|     |  |  +-- invalid-sid?     yang:counter64
|     |  |  +-- invalid-label?   yang:counter64
|     |  |  +-- tx
|     |  |     +-- packets?   yang:counter64
|     |  +-- internal
|     |     +-- packets?        yang:counter64
|     |     +-- parity-error?   yang:counter64
|     +-- policy
|     |  +-- l2
|     |  |  +-- frames?   yang:counter64
|     |  |  +-- acl?      yang:counter64
|     |  +-- l3
|     |     +-- packets?      yang:counter64
|     |     +-- acl?          yang:counter64
|     |     +-- policer
|     |     |  +-- packets?   yang:counter64
|     |     |  +-- bytes?     yang:counter64
|     |     |  +-- classes!
|     |     |     +-- class* [id]
```

```
|     |     |        +-- id         string
|     |     |        +-- packets?   yang:counter64
|     |     |        +-- bytes?     yang:counter64
|     |     +-- null-route?   yang:counter64
|     |     +-- rpf?          yang:counter64
|     |     +-- ddos?         yang:counter64
|     +-- no-buffer
|        +-- qos!
|           +-- class* [id]
|              +-- id         string
|              +-- packets?   yang:counter64
|              +-- bytes?     yang:counter64
+-- flow* [direction] {pdr-common:flow-reporting}?
|  +-- direction    identityref
|  +-- traffic
|  |  +-- l2
|  |  |  +-- frames?   yang:counter64
|  |  |  +-- bytes?    yang:counter64
|  |  +-- l3
|  |  |  +-- address-family-stat* [address-family]
|  |  |     +-- address-family    identityref
|  |  |     +-- packets?          yang:counter64
|  |  |     +-- bytes?            yang:counter64
|  |  |     +-- unicast
|  |  |     |  +-- packets?   yang:counter64
|  |  |     |  +-- bytes?     yang:counter64
|  |  |     +-- multicast
|  |  |     |  +-- packets?   yang:counter64
|  |  |     |  +-- bytes?     yang:counter64
|  |  |     +-- broadcast
|  |  |        +-- packets?   yang:counter64
|  |  |        +-- bytes?     yang:counter64
|  |  +-- qos!
|  |     +-- class* [id]
|  |        +-- id         string
|  |        +-- packets?   yang:counter64
|  |        +-- bytes?     yang:counter64
|  +-- discards
|     +-- l2
|     |  +-- frames?   yang:counter64
|     |  +-- bytes?    yang:counter64
|     +-- l3
|     |  +-- address-family-stat* [address-family]
|     |     +-- address-family    identityref
|     |     +-- packets?          yang:counter64
|     |     +-- bytes?            yang:counter64
|     |     +-- unicast
|     |     |  +-- packets?   yang:counter64
```

```
|     |     |  +-- bytes?     yang:counter64
|     |     +-- multicast
|     |     |  +-- packets?   yang:counter64
|     |     |  +-- bytes?     yang:counter64
|     |     +-- broadcast
|     |        +-- packets?   yang:counter64
|     |        +-- bytes?     yang:counter64
|     +-- errors
|     |  +-- l2
|     |  |  +-- rx
|     |  |  |  +-- frames?          yang:counter64
|     |  |  |  +-- crc-error?       yang:counter64
|     |  |  |  +-- invalid-mac?     yang:counter64
|     |  |  |  +-- invalid-vlan?    yang:counter64
|     |  |  |  +-- invalid-frame?   yang:counter64
|     |  |  +-- tx
|     |  |     +-- frames?   yang:counter64
|     |  +-- l3
|     |  |  +-- rx
|     |  |  |  +-- packets?          yang:counter64
|     |  |  |  +-- checksum-error?   yang:counter64
|     |  |  |  +-- mtu-exceeded?     yang:counter64
|     |  |  |  +-- invalid-packet?   yang:counter64
|     |  |  +-- ttl-expired?     yang:counter64
|     |  |  +-- no-route?        yang:counter64
|     |  |  +-- invalid-sid?     yang:counter64
|     |  |  +-- invalid-label?   yang:counter64
|     |  |  +-- tx
|     |  |     +-- packets?   yang:counter64
|     |  +-- internal
|     |     +-- packets?        yang:counter64
|     |     +-- parity-error?   yang:counter64
|     +-- policy
|     |  +-- l2
|     |  |  +-- frames?   yang:counter64
|     |  |  +-- acl?      yang:counter64
|     |  +-- l3
|     |     +-- packets?      yang:counter64
|     |     +-- acl?          yang:counter64
|     |     +-- policer
|     |     |  +-- packets?   yang:counter64
|     |     |  +-- bytes?     yang:counter64
|     |     |  +-- classes!
|     |     |     +-- class* [id]
|     |     |        +-- id         string
|     |     |        +-- packets?   yang:counter64
|     |     |        +-- bytes?     yang:counter64
|     |     +-- null-route?   yang:counter64
```

```
|     |     +-- rpf?          yang:counter64
|     |     +-- ddos?        yang:counter64
|     +-- no-buffer
|        +-- qos!
|           +-- class* [id]
|              +-- id        string
|              +-- packets?  yang:counter64
|              +-- bytes?    yang:counter64
+-- device {pdr-common:device-stats}?
   +-- traffic
   |  +-- l2
   |  |  +-- frames?  yang:counter64
   |  |  +-- bytes?   yang:counter64
   |  +-- l3
   |  |  +-- address-family-stat* [address-family]
   |  |     +-- address-family   identityref
   |  |     +-- packets?         yang:counter64
   |  |     +-- bytes?           yang:counter64
   |  |     +-- unicast
   |  |     |  +-- packets?  yang:counter64
   |  |     |  +-- bytes?    yang:counter64
   |  |     +-- multicast
   |  |     |  +-- packets?  yang:counter64
   |  |     |  +-- bytes?    yang:counter64
   |  |     +-- broadcast
   |  |        +-- packets?  yang:counter64
   |  |        +-- bytes?    yang:counter64
   |  +-- qos!
   |     +-- class* [id]
   |        +-- id        string
   |        +-- packets?  yang:counter64
   |        +-- bytes?    yang:counter64
   +-- discards
      +-- l2
      |  +-- frames?  yang:counter64
      |  +-- bytes?   yang:counter64
      +-- l3
      |  +-- address-family-stat* [address-family]
      |     +-- address-family   identityref
      |     +-- packets?         yang:counter64
      |     +-- bytes?           yang:counter64
      |     +-- unicast
      |     |  +-- packets?  yang:counter64
      |     |  +-- bytes?    yang:counter64
      |     +-- multicast
      |     |  +-- packets?  yang:counter64
      |     |  +-- bytes?    yang:counter64
      |     +-- broadcast
```

```
|        +-- packets?   yang:counter64
|        +-- bytes?     yang:counter64
+-- errors
|  +-- l2
|  |  +-- rx
|  |  |  +-- frames?          yang:counter64
|  |  |  +-- crc-error?       yang:counter64
|  |  |  +-- invalid-mac?     yang:counter64
|  |  |  +-- invalid-vlan?    yang:counter64
|  |  |  +-- invalid-frame?   yang:counter64
|  |  +-- tx
|  |     +-- frames?   yang:counter64
|  +-- l3
|  |  +-- rx
|  |  |  +-- packets?          yang:counter64
|  |  |  +-- checksum-error?   yang:counter64
|  |  |  +-- mtu-exceeded?     yang:counter64
|  |  |  +-- invalid-packet?   yang:counter64
|  |  +-- ttl-expired?     yang:counter64
|  |  +-- no-route?        yang:counter64
|  |  +-- invalid-sid?     yang:counter64
|  |  +-- invalid-label?   yang:counter64
|  |  +-- tx
|  |     +-- packets?   yang:counter64
|  +-- internal
|     +-- packets?        yang:counter64
|     +-- parity-error?   yang:counter64
+-- policy
|  +-- l2
|  |  +-- frames?   yang:counter64
|  |  +-- acl?      yang:counter64
|  +-- l3
|     +-- packets?      yang:counter64
|     +-- acl?          yang:counter64
|     +-- policer
|     |  +-- packets?   yang:counter64
|     |  +-- bytes?     yang:counter64
|     |  +-- classes!
|     |     +-- class* [id]
|     |        +-- id         string
|     |        +-- packets?   yang:counter64
|     |        +-- bytes?     yang:counter64
|     +-- null-route?   yang:counter64
|     +-- rpf?          yang:counter64
|     +-- ddos?         yang:counter64
+-- no-buffer
   +-- qos!
      +-- class* [id]
```

```
                     +-- id         string
                     +-- packets?   yang:counter64
                     +-- bytes?     yang:counter64
```

Appendix D. Full Data Model Tree

The following YANG tree diagram shows the complete DM structure:

```
module: ietf-packet-discard-reporting

  augment /rt:routing/rt:control-plane-protocols
            /rt:control-plane-protocol:
    +--ro discard-stats {pdr-common:control-plane-stats}?
       +--ro discard-order-capability*   identityref
       +--ro traffic* [direction]
       |  +--ro direction    identityref
       |  +--ro packets?     yang:counter64
       |  +--ro bytes?       yang:counter64
       +--ro discards* [direction]
          +--ro direction    identityref
          +--ro packets?     yang:counter64
          +--ro bytes?       yang:counter64
          +--ro policy
             +--ro packets?   yang:counter64
  augment /if:interfaces/if:interface/if:statistics:
    +--ro discard-order-capability*   identityref
    |       {pdr-common:interface-stats}?
    +--ro traffic* [direction] {pdr-common:interface-stats}?
    |  +--ro direction    identityref
    |  +--ro l2
    |  |  +--ro frames?   yang:counter64
    |  |  +--ro bytes?    yang:counter64
    |  +--ro l3
    |  |  +--ro address-family-stat* [address-family]
    |  |     +--ro address-family    identityref
    |  |     +--ro packets?          yang:counter64
    |  |     +--ro bytes?            yang:counter64
    |  |     +--ro unicast
    |  |     |  +--ro packets?   yang:counter64
    |  |     |  +--ro bytes?     yang:counter64
    |  |     +--ro multicast
    |  |     |  +--ro packets?   yang:counter64
    |  |     |  +--ro bytes?     yang:counter64
    |  |     +--ro broadcast
    |  |        +--ro packets?   yang:counter64
    |  |        +--ro bytes?     yang:counter64
    |  +--ro qos!
    |     +--ro class* [id]
```

```
|        +--ro id         string
|        +--ro packets?   yang:counter64
|        +--ro bytes?     yang:counter64
+--ro discards* [direction] {pdr-common:interface-stats}?
   +--ro direction    identityref
   +--ro l2
   |  +--ro frames?   yang:counter64
   |  +--ro bytes?    yang:counter64
   +--ro l3
   |  +--ro address-family-stat* [address-family]
   |     +--ro address-family    identityref
   |     +--ro packets?          yang:counter64
   |     +--ro bytes?            yang:counter64
   |     +--ro unicast
   |     |  +--ro packets?   yang:counter64
   |     |  +--ro bytes?     yang:counter64
   |     +--ro multicast
   |     |  +--ro packets?   yang:counter64
   |     |  +--ro bytes?     yang:counter64
   |     +--ro broadcast
   |        +--ro packets?   yang:counter64
   |        +--ro bytes?     yang:counter64
   +--ro errors
   |  +--ro l2
   |  |  +--ro rx
   |  |  |  +--ro frames?          yang:counter64
   |  |  |  +--ro crc-error?       yang:counter64
   |  |  |  +--ro invalid-mac?     yang:counter64
   |  |  |  +--ro invalid-vlan?    yang:counter64
   |  |  |  +--ro invalid-frame?   yang:counter64
   |  |  +--ro tx
   |  |     +--ro frames?   yang:counter64
   |  +--ro l3
   |  |  +--ro rx
   |  |  |  +--ro packets?          yang:counter64
   |  |  |  +--ro checksum-error?   yang:counter64
   |  |  |  +--ro mtu-exceeded?     yang:counter64
   |  |  |  +--ro invalid-packet?   yang:counter64
   |  |  +--ro ttl-expired?     yang:counter64
   |  |  +--ro no-route?        yang:counter64
   |  |  +--ro invalid-sid?     yang:counter64
   |  |  +--ro invalid-label?   yang:counter64
   |  |  +--ro tx
   |  |     +--ro packets?   yang:counter64
   |  +--ro internal
   |     +--ro packets?        yang:counter64
   |     +--ro parity-error?   yang:counter64
   +--ro policy
```

```
         |  +--ro l2
         |  |  +--ro frames?   yang:counter64
         |  |  +--ro acl?      yang:counter64
         |  +--ro l3
         |     +--ro packets?      yang:counter64
         |     +--ro acl?          yang:counter64
         |     +--ro policer
         |     |  +--ro packets?   yang:counter64
         |     |  +--ro bytes?     yang:counter64
         |     |  +--ro classes!
         |     |     +--ro class* [id]
         |     |        +--ro id         string
         |     |        +--ro packets?   yang:counter64
         |     |        +--ro bytes?     yang:counter64
         |     +--ro null-route?   yang:counter64
         |     +--ro rpf?          yang:counter64
         |     +--ro ddos?         yang:counter64
         +--ro no-buffer
            +--ro qos!
               +--ro class* [id]
                  +--ro id         string
                  +--ro packets?   yang:counter64
                  +--ro bytes?     yang:counter64
     augment /lne:logical-network-elements/lne:logical-network-element:
       +--ro discard-stats {pdr-common:device-stats}?
          +--ro discard-order-capability*   identityref
          +--ro traffic
          |  +--ro l2
          |  |  +--ro frames?   yang:counter64
          |  |  +--ro bytes?    yang:counter64
          |  +--ro l3
          |  |  +--ro address-family-stat* [address-family]
          |  |     +--ro address-family    identityref
          |  |     +--ro packets?          yang:counter64
          |  |     +--ro bytes?            yang:counter64
          |  |     +--ro unicast
          |  |     |  +--ro packets?   yang:counter64
          |  |     |  +--ro bytes?     yang:counter64
          |  |     +--ro multicast
          |  |     |  +--ro packets?   yang:counter64
          |  |     |  +--ro bytes?     yang:counter64
          |  |     +--ro broadcast
          |  |        +--ro packets?   yang:counter64
          |  |        +--ro bytes?     yang:counter64
          |  +--ro qos!
          |     +--ro class* [id]
          |        +--ro id         string
          |        +--ro packets?   yang:counter64
```

```
|        +--ro bytes?     yang:counter64
+--ro discards
   +--ro l2
   |  +--ro frames?   yang:counter64
   |  +--ro bytes?    yang:counter64
   +--ro l3
   |  +--ro address-family-stat* [address-family]
   |     +--ro address-family    identityref
   |     +--ro packets?          yang:counter64
   |     +--ro bytes?            yang:counter64
   |     +--ro unicast
   |     |  +--ro packets?   yang:counter64
   |     |  +--ro bytes?     yang:counter64
   |     +--ro multicast
   |     |  +--ro packets?   yang:counter64
   |     |  +--ro bytes?     yang:counter64
   |     +--ro broadcast
   |        +--ro packets?   yang:counter64
   |        +--ro bytes?     yang:counter64
   +--ro errors
   |  +--ro l2
   |  |  +--ro rx
   |  |  |  +--ro frames?          yang:counter64
   |  |  |  +--ro crc-error?       yang:counter64
   |  |  |  +--ro invalid-mac?     yang:counter64
   |  |  |  +--ro invalid-vlan?    yang:counter64
   |  |  |  +--ro invalid-frame?   yang:counter64
   |  |  +--ro tx
   |  |     +--ro frames?   yang:counter64
   |  +--ro l3
   |  |  +--ro rx
   |  |  |  +--ro packets?          yang:counter64
   |  |  |  +--ro checksum-error?   yang:counter64
   |  |  |  +--ro mtu-exceeded?     yang:counter64
   |  |  |  +--ro invalid-packet?   yang:counter64
   |  |  +--ro ttl-expired?     yang:counter64
   |  |  +--ro no-route?        yang:counter64
   |  |  +--ro invalid-sid?     yang:counter64
   |  |  +--ro invalid-label?   yang:counter64
   |  |  +--ro tx
   |  |     +--ro packets?   yang:counter64
   |  +--ro internal
   |     +--ro packets?        yang:counter64
   |     +--ro parity-error?   yang:counter64
   +--ro policy
   |  +--ro l2
   |  |  +--ro frames?   yang:counter64
   |  |  +--ro acl?      yang:counter64
```

```
|  +--ro l3
|     +--ro packets?      yang:counter64
|     +--ro acl?          yang:counter64
|     +--ro policer
|     |  +--ro packets?   yang:counter64
|     |  +--ro bytes?     yang:counter64
|     |  +--ro classes!
|     |     +--ro class* [id]
|     |        +--ro id         string
|     |        +--ro packets?   yang:counter64
|     |        +--ro bytes?     yang:counter64
|     +--ro null-route?   yang:counter64
|     +--ro rpf?          yang:counter64
|     +--ro ddos?         yang:counter64
+--ro no-buffer
   +--ro qos!
      +--ro class* [id]
         +--ro id         string
         +--ro packets?   yang:counter64
         +--ro bytes?     yang:counter64
```

Acknowledgements

The content of this document has benefitted from feedback from JR Rivers, Ronan Waide, Chris DeBruin, and Marcos Sanz.

Thanks to Benoテョt Claise, Joe Clarke, Tom Petch, Mahesh Jethanandani, Paul Aitken, and Randy Bush for the review and comments.

Thanks to Ladislav Lhotka for the YANGDOCTORS reviews, Sergio Belotti for the OPSDIR review, and Satoru Matsushima for the INTDIR review.

Thanks to Diego Lopez for shepherding the document and Mahesh Jethanandani for the AD review.

Contributors

Nadav Chachmon
Cisco Systems, Inc.
170 West Tasman Dr.
San Jose, CA 95134
United States of America
Email: nchachmo@cisco.com

Authors' Addresses

John Evans (editor)
Amazon
1 Principal Place, Worship Street
London
EC2A 2FA
United Kingdom
Email: jevanamz@amazon.co.uk

Oleksandr Pylypenko (editor)
Nvidia
2788 San Tomas Expy
Santa Clara, CA 95051
United States of America
Email: opylypenko@nvidia.com

Jeffrey Haas
Juniper Networks
1133 Innovation Way
Sunnyvale, CA 94089
United States of America
Email: jhaas@juniper.net

Aviran Kadosh
Cisco Systems, Inc.
170 West Tasman Dr.
San Jose, CA 95134
United States of America
Email: akadosh@cisco.com

Mohamed Boucadair (editor)
Orange
France
Email: mohamed.boucadair@orange.com