OPSAWG
Internet-Draft
Intended status: Informational
Expires: 22 March 2026

J. Evans
O. Pylypenko
K. Cheaito
Amazon

18 September 2025

Information Element for Flow Discard Classification
draft-evans-opsawg-ipfix-discard-class-ie-01

Abstract

This document defines an IPFIX Information Element for classifying flow-level discards that aligns with the information model defined in [I-D.ietf-opsawg-discardmodel]. The flowDiscardClass Information Element provides consistent classification of packet discards across IPFIX implementations, enabling correlation between device, interface and control-plane discards and the impacted flows.

Table of Contents

1. Introduction

For network operators, understanding both where and why packet loss occurs within a network is essential for effective operation. 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.

[I-D.ietf-opsawg-discardmodel] addresses this need by defining an information model that provides precise classification of packet loss, enabling accurate automated mitigation. While its YANG data model implementation provides device, interface and control-plane discards, effective automated triage often requires understanding which specific flows are impacted. For example, when mitigating congestion, operators may need to identify and trace the sources of elephant flows. This requires the ability to correlate device and interface-level discard classes with the specific flows being dropped.

Currently, [RFC7270] defines the forwardingStatus Information Element for reporting packet forwarding outcomes in IPFIX, including various reasons for packet drops. The defined drop reason codes lack the granularity and clarity needed for automated root cause analysis and impact mitigation, however. For instance, the "For us" reason code provides insufficient context to determine appropriate mitigation actions.

This document addresses these limitations by introducing a new Information Element, flowDiscardClass, to provide a consistent classification scheme for packet discards across IPFIX implementations. This new element aligns with the classification scheme defined in [I-D.ietf-opsawg-discardmodel] and enables:

1. Precise detection of unintended packet loss through clear distinction between intended and unintended discards

2. Accurate root cause analysis through detailed classification of discard reasons

3. Automated selection of mitigation actions based on discard type, rate, and duration

4. Consistent reporting across vendor implementations in both YANG and IPFIX data models

By providing this mapping between YANG and IPFIX implementations, this document enables operators to correlate device-level statistics with flow-level impacts, facilitating more effective automated network operations.

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.

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

Intended discards are packets dropped due to deliberate network policies or configurations designed to enforce security or quality of service. For example, packets dropped because they match an Access Control List (ACL) denying certain traffic types.

Unintended discards 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. Information Element

This Information Element has been specified in accordance with the guidelines in [RFC7013].

3.1. Design Rationale

The mapping between [I-D.ietf-opsawg-discardmodel] and the IPFIX flowDiscardClass Information Element follows these principles, maintaining consistency with the YANG model while allowing self-contained decoding from a single IE:

1. Scope. The flowDiscardClass Information Element is specifically for reporting flow-level discard reasons, and therefore only represents the flow subtree from [I-D.ietf-opsawg-discardmodel].

The component is implicitly "flow" and the type is implicitly "discards"; interface, device, and control-plane components are out of scope for this IE.

2. Hierarchy preserved. The enumeration mirrors the model: both leaves (specific reasons) and structural aggregates are assigned values so collectors can perform coarse or fine roll-ups. For L3, structural aggregates include address-family and cast (v4/v6, unicast/multicast).

3. Self-contained decoding. The value alone carries the discard class. Exporters and collectors can still use other IEs (e.g., flowDirection, ipVersion, addresses, ipDiffServCodePoint) for correlation, but they are not required to decode the class.

4. Specificity preference. The scheme encourages reporting the most-specific known class when available; aggregate values provide a fallback when only a broader category is known.

5. Implementation-friendly ordering. Codes are assigned in preorder (depth-first) tree order to reflect the hierarchy and simplify range/roll-up handling in implementations.

3.2. flowDiscardClass Definition

Name: flowDiscardClass

Description: Classifies the reason a packet was discarded in a flow, using the hierarchical classification scheme defined in [I-D.ietf-opsawg-discardmodel].

Abstract Data Type: unsigned8

Data Type Semantics: identifier

Units: none

Range: 0..38 (values from Table 1; other values are unassigned and MUST be treated as unknown)

Reversibility: reversible (value does not change under flow reversal as per [RFC5103])

Status: current

ElementId: TBD

References: [I-D.ietf-opsawg-discardmodel]

3.3. flowDiscardClass Values

Table 1 defines the values for the flowDiscardClass Information Element mapped from the corresponding [I-D.ietf-opsawg-discardmodel] Discard Class. The code points for flowDiscardClass are maintained by IANA in the "flowDiscardClass Values" subregistry of the IPFIX registry. Future additions or changes are managed via Expert Review as described in Section 5.

Discard Class	flowDiscardClass Value
l2	0
l3	1
l3/v4	2
l3/v4/unicast	3
l3/v4/multicast	4
l3/v6	5
l3/v6/unicast	6
l3/v6/multicast	7
errors	8
errors/l2	9
errors/l2/rx	10
errors/l2/rx/crc-error	11
errors/l2/rx/invalid-mac	12
errors/l2/rx/invalid-vlan	13
errors/l2/rx/invalid-frame	14
errors/l2/tx	15
errors/l3	16
errors/l3/rx	17

errors/l3/rx/checksum-error	18
errors/l3/rx/mtu-exceeded	19
errors/l3/rx/invalid-packet	20
errors/l3/ttl-expired	21
errors/l3/no-route	22
errors/l3/invalid-sid	23
errors/l3/invalid-label	24
errors/l3/tx	25
errors/internal	26
errors/internal/parity-error	27
policy	28
policy/l2	29
policy/l2/acl	30
policy/l3	31
policy/l3/acl	32
policy/l3/policer	33
policy/l3/null-route	34
policy/l3/rpf	35
policy/l3/ddos	36
no-buffer	37
no-buffer/class	38

Table 1: Flow discard classification values and corresponding discard classes

Codes are assigned in preorder (depth-first) tree order to reflect the model's hierarchy.

no-buffer/class conveys per-QoS class congestion loss; the specific class (e.g., DSCP/class index, or L2 PCP) SHOULD be exported via the appropriate companion IE in the same record.

3.4. Implementation Requirements

Semantics and Scope {#impl-semantics}

1. Scope of this IE. flowDiscardClass MUST be used only to report flow-level discard classification under flow/discards from [I-D.ietf-opsawg-discardmodel]. It MUST NOT be used for interface, device, or control-plane discard counters.

2. Enumeration. Exporters MUST encode values only from the IANA “flowDiscardClass Values” subregistry for this IE. Collectors MUST accept both aggregate and leaf values and interpret aggregates as semantic supersets of their descendants.

3. Unknown/Unassigned values. Collectors receiving an unknown or unassigned value MUST treat it as unknown and MUST NOT remap it to another code. Exporters MUST NOT transmit unassigned values.

4. Reversibility. The value of flowDiscardClass MUST NOT change under biflow reversal as defined by [RFC5103].

3.4.1. Exporter Requirements

1. Cardinality. A Flow Record MUST contain at most one instance of flowDiscardClass.

2. Multiplicity. When multiple discard reasons apply to the same flow interval, exporters MUST either (a) export multiple Flow Records (one per reason) or (b) encode a basicList of flowDiscardClass values using IPFIX Structured Data per [RFC6313]/[RFC7013].

3. Specificity. Exporters SHOULD report the most-specific known class (a leaf). If the specific leaf is unknown, an appropriate parent/aggregate MAY be used.

4. Interval semantics. When exported on an interval Flow Record, the presence of flowDiscardClass indicates that at least one packet in the interval matched that class. Exporters SHOULD include droppedPacketDeltaCount and/or droppedOctetDeltaCount in the same record to quantify the affected volume.

5. Congestive loss traffic class (no-buffer/class).

* If flowDiscardClass equals no-buffer/class, the traffic-class identifier used by the queueing system MUST be present in the same record, carried in exactly one suitable IE (e.g., ipDiffServCodePoint or ipClassOfService for L3, or dot1qPriority for L2).

* If classification occurs after remarking, exporters MUST use the corresponding post-class IE where available, or provide a device queue-ID→class mapping via IPFIX Options data.

6. Context (recommended). To aid correlation with interface/device/control-plane counters, exporters SHOULD include time bounds (flowStart/flowEnd or an observation-time IE), ingressInterface/egressInterface as applicable, and observationPointId when multiple pipeline stages/taps exist.

3.4.2. Collector Requirements

1. Lists and duplicates. Collectors MUST be able to parse a basicList of flowDiscardClass values per [RFC6313]. If multiple Flow Records carry different flowDiscardClass values for the same flow keys/time bucket, collectors MAY treat them as separate reasons for analysis.

2. Aggregate handling. When a parent/aggregate class is received, collectors MUST treat it as a coarse classification that may encompass multiple leaves.

3. Congestive loss traffic class. For no-buffer/class, when a traffic-class IE is present, collectors MUST use it to align with per-class counters; if absent, collectors MAY apply local device mappings if available.

4. Unknown values. Collectors MUST handle unknown/unassigned values gracefully (e.g., categorize as “unknown”) without rejecting the record.

3.4.3. Interoperability with Existing IPFIX IEs

1. Exporters and collectors MAY also use existing IEs (e.g., flowDirection, ipVersion, addresses, ipDiffServCodePoint) for filtering, correlation, or redundancy.

2. flowDiscardClass alone SHOULD be sufficient to recover the discard classification (apart from the traffic-class identity required for no-buffer/class above).

3. Exporters MAY continue to export forwardingStatus ([RFC7270]) in parallel. When both are present, flowDiscardClass SHOULD be considered authoritative for discard classification.

4. When flow sampling is active, the presence of flowDiscardClass indicates at least one sampled packet matched that class.

4. Security Considerations

This document defines a new Information Element for flow-level discard classification to align with the information model defined in [I-D.ietf-opsawg-discardmodel]. As such, there are no security issues related to this document, which are additional to those discussed in [RFC7011], [RFC7012].

5. IANA Considerations

IANA is requested to make the following changes under the IP Flow Information Export (IPFIX) Information Elements registry.

5.1. New IPFIX Information Element: flowDiscardClass

IANA is requested to register a new Information Element as follows:

* Name: flowDiscardClass

* ElementId: TBD (to be assigned by IANA)

* Description: Classifies the reason a packet was discarded in a flow, using the hierarchical classification scheme defined in [I-D.ietf-opsawg-discardmodel].

* Abstract Data Type: unsigned8

* Data Type Semantics: identifier

* Units: none

* Range: 0..38 (values are listed in the “flowDiscardClass Values” subregistry created below; other values are unassigned and MUST be treated as unknown)

* Reversibility: reversible (value does not change under flow reversal as per [RFC5103])

* Status: current

* Reference: This document; [RFC7013]

5.2. New Subregistry: “flowDiscardClass Values”

IANA is requested to create a new subregistry titled "flowDiscardClass Values" under the IPFIX Information Elements registry. This subregistry contains the enumerated values for the flowDiscardClass IE.

* Registration Procedure: Expert Review ([RFC8126])

* Reference: This document; [RFC7013]

* Fields:

  - Value (integer)

  - Name (path under flow/discards/...)

  - Description (optional)

  - Reference

Designated Expert guidance: New code points should reflect additions to or clarifications of discard reasons in [I-D.ietf-opsawg-discardmodel] (or its successor). Existing code points MUST NOT be repurposed. Backwards-compatible additions are preferred. Experts SHOULD maintain the hierarchical structure (e.g., assigning aggregates and leaves consistently) and, where practical, preserve preorder (depth-first) numbering to align with the existing tree.

6. References

6.1. Normative References

[I-D.ietf-opsawg-discardmodel]
Evans, J., Pylypenko, O., Haas, J., Kadosh, A., and M. Boucadair, "Information and Data Models for Packet Discard Reporting", Work in Progress, Internet-Draft, draft-ietf-opsawg-discardmodel-09, 18 September 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-opsawg-discardmodel-09>.

[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>.

[RFC5103] Trammell, B. and E. Boschi, "Bidirectional Flow Export Using IP Flow Information Export (IPFIX)", RFC 5103, DOI 10.17487/RFC5103, January 2008, <https://www.rfc-editor.org/rfc/rfc5103>.

[RFC6313] Claise, B., Dhandapani, G., Aitken, P., and S. Yates, "Export of Structured Data in IP Flow Information Export (IPFIX)", RFC 6313, DOI 10.17487/RFC6313, July 2011, <https://www.rfc-editor.org/rfc/rfc6313>.

[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>.

[RFC7012] Claise, B., Ed. and B. Trammell, Ed., "Information Model for IP Flow Information Export (IPFIX)", RFC 7012, DOI 10.17487/RFC7012, September 2013, <https://www.rfc-editor.org/rfc/rfc7012>.

[RFC7013] Trammell, B. and B. Claise, "Guidelines for Authors and Reviewers of IP Flow Information Export (IPFIX) Information Elements", BCP 184, RFC 7013, DOI 10.17487/RFC7013, September 2013, <https://www.rfc-editor.org/rfc/rfc7013>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/rfc/rfc8126>.

[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>.

6.2. Informative References

[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>.

Appendix A. Correlating Flow Discards with Interface/Device/Control-Plane Discards

Objective. Enable operators to understand which flows are impacted when interface, device, or control-plane discard counters rise in the discardmodel. A typical workflow is:

1. Detect anomalous discards on an interface/device/control-plane (by class and direction) using the discardmodel; then

2. Query flow telemetry to identify the impacted flows (and, where applicable, the causal flows that contributed to the condition).

A.1. Scope Alignment (What Must Match)

Accurate correlation depends on joining records that describe the same exporter/vantage, component, interface (if applicable), direction, class, and time window:

* Exporter & vantage. Join on the same Exporter/Observation Domain (e.g., observationDomainId). Where multiple taps or pipeline stages exist, include observationPointId (and, if available, line-card/port identifiers) so flow data and counters represent comparable points.

* Component & interface.

  - Interface component: include ingressInterface/egressInterface in Flow Records and match the same ifIndex and direction as the discardmodel counters.

  - Device component: drop the interface key and aggregate across interfaces for the same exporter/time/class.

  - Control-plane component: no interface key; match by exporter/time/class (and any control-plane class identifiers available).

* Time. Align Flow intervals (flowStart/flowEnd, or an observation-time IE) with the sampling/roll-up interval of the discardmodel counters. In practice, bucket both datasets (e.g., 1-minute buckets) and allow small skew to cover clock/polling jitter.

A.2. Class and Reason Alignment

* Discard class. flowDiscardClass mirrors the discardmodel hierarchy (parents + leaves). Select the value(s) that match the anomaly (e.g., errors/l3/ttl-expired, policy/l3/acl, no-buffer/class, etc.).

* Traffic class identity (when relevant). For no-buffer/class, Exporters SHOULD also export the traffic-class identifier used by the queue (e.g., ipDiffServCodePoint / ipClassOfService, or dot1qPriority for L2). If the device uses internal queue IDs, provide a mapping (via Options data or out-of-band config) so collectors can align Flow Records with per-class counters.

A.3. Recommended Exporter Context

When exporting Flow Records that carry flowDiscardClass, Exporters SHOULD include:

* Context: ingressInterface/egressInterface, observationPointId (if applicable), time bounds (flowStart/flowEnd or an observation-time IE), and the relevant class IE (ipDiffServCodePoint/ipClassOfService/dot1qPriority, etc.).

* Quantification: droppedPacketDeltaCount and/or droppedOctetDeltaCount, so per-class dropped volume from flows can be compared to discardmodel aggregates.

* Multiplicity handling: if multiple discard reasons apply to the same flow interval, either export one record per reason or use IPFIX Structured Data to encode multiple flowDiscardClass values.

A.4. Collector Workflow (General Pattern)

Given an anomaly in the discardmodel (per interface/device/control-plane, direction, class, and time):

1. Select the time bucket(s) and affected key(s): exporter, component (and interface if applicable), direction, discard class.

2. Find impacted flows: filter Flow Records that overlap the bucket(s) and match the exporter/component keys and either

    * explicitly report the same flowDiscardClass, or

    * (for aggregate classes) belong to the same traffic class and vantage where the loss is counted.

3. Aggregate per flow within the bucket(s): bytes/packets and, if available, dropped-bytes/packets.

4. Rank or group flows as needed:

   * Impacted analysis: which flows suffered loss (by dropped-octets/packets, or by presence of the discard class)?

   * Causal analysis (when meaningful): which flows likely contributed to the interface/device condition (e.g., top senders in the same class and egress interface during a no-buffer/class spike)?

5. Validate: compare summed flow-level dropped deltas (if exported) to the discardmodel deltas for that bucket. Small gaps are expected (sampling, timing, vantage); large gaps suggest vantage mismatch or incomplete class mapping.

A.5. Handling Expected Discrepancies

* Vantage mismatch. Flow telemetry captured before queueing/policing but counters tallied after will skew attribution. Use observationPointId (and device documentation) to align vantage.

* Class remapping. If DSCP is remarked, use post-class IEs or queue-ID mappings so class identity matches where the drop is counted.

* Sampling/filtering. Flow sampling reduces visibility of small flows and biases shares; where possible, rely on flow-level dropped-octet counters, or increase the bucket size to stabilize estimates.

* Clock skew. Apply a small skew window (e.g., ±30 s) when joining buckets across datasets.

A.6. Operational Example: Congestive Loss (no-buffer/class) and Elephant Flows

This example illustrates the workflow above for congestive loss (no-buffer/class) on an egress interface. It identifies high-volume ("elephant") flows most likely responsible for a spike by joining per-class interface discards with per-class, per-interface flow aggregates and ranking flows by bytes/rate.

Assumed tables:

* flows(observation_domain_id, egress_ifindex, flow_start, flow_end, octet_delta, packet_delta, ip_dscp, src_addr, dst_addr, src_port, dst_port, protocol, flowdiscardclass)

* interface_discards(observation_domain_id, ifindex, direction, discard_class, class_id, ts, packet_delta, octet_delta)

Note: In Table 1, no-buffer/class has value 38.

```
-- Identify elephant flows contributing to egress no-buffer/class drops
WITH params AS (
  SELECT
    38::smallint AS no_buffer_class,          -- flowDiscardClass for no-buffer/class
(Table 1)
    interval '1 minute' AS bucket,            -- analysis granularity
    interval '30 seconds' AS skew,            -- clock/bucket tolerance
    100000000::bigint AS elephant_bytes_min   -- 100 MB per bucket (example threshold)
),

-- 1) Per-minute egress no-buffer/class discard spikes per interface + class
events AS (
  SELECT
      i.observation_domain_id,
      i.ifindex,
      i.class_id,                               -- DSCP / QoS class as exported by the
 device
      date_trunc('minute', i.ts) AS ts_bucket,
      SUM(i.packet_delta) AS drop_pkts,
      SUM(i.octet_delta)  AS drop_octets
  FROM interface_discards i
  JOIN params p ON i.discard_class = p.no_buffer_class
  WHERE i.direction = 'egress'
  GROUP BY 1,2,3,4
  HAVING SUM(i.packet_delta) > 0
),

-- 2) Aggregate flows per minute keyed by egress interface + class
flow_buckets AS (
  SELECT
      f.observation_domain_id,
      f.egress_ifindex AS ifindex,
      f.ip_dscp        AS class_id,
      date_trunc('minute', f.flow_end) AS ts_bucket,
      f.src_addr, f.dst_addr, f.src_port, f.dst_port, f.protocol,
      SUM(f.octet_delta)  AS bytes,
      SUM(f.packet_delta) AS pkts
  FROM flows f
  -- Optional: uncomment to include only flows explicitly marked as no-buffer/class
  -- WHERE f.flowdiscardclass = 38
  GROUP BY 1,2,3,4,5,6,7,8,9
```

```
   ),

   -- 3) Join flows to discard spikes within a fuzzy time window and same class/interface
   joined AS (
     SELECT
         e.observation_domain_id,
         e.ifindex,
         e.class_id,
         e.ts_bucket,
         e.drop_pkts,
         e.drop_octets,
         fb.src_addr, fb.dst_addr, fb.src_port, fb.dst_port, fb.protocol,
         fb.bytes, fb.pkts,
         (fb.bytes::numeric / NULLIF(e.drop_octets,0)) AS byte_share,
         (fb.pkts::numeric  / NULLIF(e.drop_pkts,0))   AS pkt_share
     FROM events e
     JOIN flow_buckets fb
       ON fb.observation_domain_id = e.observation_domain_id
      AND fb.ifindex               = e.ifindex
      AND fb.class_id              = e.class_id
      AND fb.ts_bucket BETWEEN e.ts_bucket - (SELECT skew FROM params)
                           AND     e.ts_bucket + (SELECT bucket FROM params) + (SELECT sk
ew FROM params)
   )

   -- 4) Rank top flows per interface+class+minute and keep "elephants"
   SELECT
       observation_domain_id, ifindex, class_id, ts_bucket,
       drop_pkts, drop_octets,
       src_addr, dst_addr, src_port, dst_port, protocol,
       bytes, pkts, byte_share, pkt_share,
       (8.0 * bytes) / EXTRACT(EPOCH FROM (SELECT bucket FROM params)) AS bits_per_sec,
       ROW_NUMBER() OVER (
         PARTITION BY observation_domain_id, ifindex, class_id, ts_bucket
         ORDER BY bytes DESC
       ) AS rank_in_bucket
   FROM joined
   WHERE bytes >= (SELECT elephant_bytes_min FROM params)
   ORDER BY ts_bucket, observation_domain_id, ifindex, class_id, rank_in_bucket;
```

Implementation notes:

* If your device maps DSCP→queue differently, join via a mapping table instead of class_id = ip_dscp.

* Adjust bucket, skew, and elephant_bytes_min for your polling cadence and "elephant" threshold.

* Helpful indexes:

- interface_discards(discard_class, direction, ts, observation_domain_id, ifindex, class_id)

- flows(egress_ifindex, ip_dscp, flow_end, observation_domain_id)

Authors' Addresses

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

Oleksandr Pylypenko
Amazon
410 Terry Ave N
Seattle, WA 98109
United States of America
Email: opyl@amazon.com

Karim Cheaito
Amazon
410 Terry Ave N
Seattle, WA 98109
United States of America
Email: kcheaito@amazon.com