IPFIX Working Group
Internet Draft
Expiration Date: April 2004

EDITORS: Ganesh Sadasivan
Cisco Systems, Inc.
Nevil Brownlee
CAIDA | U Auckland
October 2003

Architecture Model for IP Flow Information Export

draft-ietf-ipfix-arch-02.txt

Status of this Memo

Abstract

This memo defines the architecture for the export of measured IP flow information out of an IPFIX device to a Collector, per the requirements defined in [IPFIX-REQS].

Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

Table of Contents

1 Introduction 3
2 Scope 3
3 Terminology 3
4 IPFIX reference Model 8
5 IPFIX Functional and Logical blocks 10
5.1 Metering Process Functions 10
5.1.1 Flow Classification 10
5.1.2 Selection Criteria for Packets 10
5.1.3 Function on properties that determines a flow type (Fi) 11
5.1.4 Sampling packets on a flow type (Si) 11
5.2 Observation Domain 12
5.3 Flow Recording Process 12
5.4 Exporting Process 12
5.5 IPFIX protocol 13
6 Encoding Control Information 13
7 Encoding Flow Data Information 14
8 Exporting Control Information 14
9 Flow Expiration and Export 15
10 Export Error Handling 16
10.1 Selection Criteria of flows for export 16
11 The Selected IPFIX Protocol 16
12 Collecting Process 20
12.1 IPFIX Protocol on Collecting Process 20
12.2 Support for Applications 20
12.3 Export Models 21
12.3.1 Export Model with Reliable Control Connection 21
12.4 Collector Crash Detection and Recovery 21
12.4.1 Export Model with Reliable Control Connection 22
12.5 Collector Redundancy 22
13 IPFIX flow collection from Special Devices 22
14 IPFIX flow collection for special traffic 23
15 Security Considerations 23
15.1 Data security 23
15.1.1 No security 24
15.1.2 Authentication only 24
15.1.3 Encryption 24
15.2 IPFIX end point authentication 25
16 IPFIX overload 25
16.1 Denial of service (DoS) attack prevention 25
16.1.1 Network under attack 25
16.1.2 Generic DoS attack on the IPFIX system 26
16.1.3 IPFIX Specific DoS attack 26
17 IANA Considerations 26
18 References 26

19 Acknowledgements 27
20 Author's Addresses 27
21 Full Copyright Statement 28

1. Introduction

There are several applications e.g., Usage-based Accounting, Traffic Profiling, Traffic engineering, Attack/Intrusion Detection, QoS Monitoring, that require flow-based IP traffic measurements. It is hence important to have a standard way of exporting information related to IP flows. This document defines architecture for IP traffic flow monitoring, measuring and exporting. It provides a high-level description of the key components and their functions.

2. Scope

This document defines architecture for IPFIX. The main objective of this document is to:

* Describe the key architectural components of IPFIX.
* Define the architectural requirements, e.g., Recovery, Security, etc. for the IPFIX framework.
* Describe the characteristics of IPFIX protocol.

3. Terminology

* Observation Point:
  The Observation Point is a location in the network where IP packets can be observed. Examples are, a line to which a probe is attached, a shared medium, such as an Ethernet-based LAN, a single port of a router, or a set of interfaces (physical or logical) of a router.

  Note that one Observation Point may be a superset of several other Observation Points. For example one Observation Point can be an entire line card. This would be the superset of the individual Observation Points at the line card's interfaces.

* IP Traffic Flow or Flow:
  A Flow is defined as a set of IP packets passing an Observation Point in a network during a certain time interval. All packets that belong to a particular Flow have a set of common properties

derived from the data contained in the packet and from the packet treatment at the Observation Point.

In the context of IPFIX a Flow is defined as follows:

A 'Flow' is a set of IP packets, or encapsulated IP packets, passing an Observation Point in the network during a certain time interval. All packets belonging to a particular Flow have a set of common properties. Each property is defined as the result of applying a function to the values of:

1. One or more header fields of the actual packet, e.g. destination IP address, or fields in the packet's encapsulation header, e.g. label for MPLS, tunnel end-points for IP-in-IP or fields in transport header (e.g. destination port number), or fields in application header field (e.g. RTP header fields [RFC1889])
2. One or more properties of the packet itself (e.g. packet length)
3. One or more of fields derived from packet treatment (e.g. next hop IP address,AS number)

A packet is defined to belong to a Flow if it completely satisfies all the defined properties of the Flow. Each of the fields from 1., 2. and 3. mentioned above are referred to as Flow Keys or Keys. This definition covers the range from a Flow containing all packets observed at a network interface to a Flow consisting of just a single packet between two applications with a specific sequence number. Note that the Flow definition does not necessarily match a general application-level end-to-end stream. Some examples of flows are listed below:

Example 1: To create flows, the different fields to distinguish flows are defined. The different combination of the field values creates unique flows. If the keys are defined as {source IP address, destination IP address, DSCP}, then all of these are different flows.

1. {192.1.40.1, 171.6.23.5, 4}
2. {192.1.40.23, 171.6.23.67, 4}
3. {192.1.40.23, 171.6.23.67, 2}
4. {198.20.9.200, 171.6.23.67, 4}

Example 2: To create flows, a match function can be applied to all the packets that pass through an Observation Point, in order to aggregate some values. This could be done by defining the keys as {source IP address, destination IP address, TOS} as in the example 1, and applying the function which masks the least

significant 8 bits of the source IP address and destination IP address (i.e. the result is a /24 address). The 4 flows from example 1 would now be aggregated into 3 flows, by merging the flows 1. and 2. into a single flow.

1. {192.1.40.0/24, 171.6.23.0/24, 4}
2. {192.1.40.0/24, 171.6.23.0/24, 2}
3. {198.20.9.0/24, 171.6.23.0/24, 4}

Example 3: To create flows, a filter defined by some field values can be applied on all packets that pass the Observation Point, in order to select only certain flows. The filter is defined by choosing fixed values for specific fields from the packet.

All the packets that go from a customer network 192.1.40.0/24 to another customer network 171.6.23.0/24 with TOS value of 4 define a flow. All other combinations don't define a flow and are not taken into account. The 3 flows from example 2 would now be reduced to 1 flow, by filtering away the second and the third flow. {192.1.40.0/24, 171.6.23.0/24, 4}.

The above example can be thought of as a function F taking as input {source IP address, destination IP address, TOS}. The function selects only the packets which satisfy all the 3 conditions which are:

* mask out the least significant 8 bits of source IP address, compare against 192.1.40.0.
* mask out the least significant 8 bits of destination IP address, compare against 171.6.23.0.
* tos value equal to 4.

Depending on the values of {source IP address, destination IP address, TOS} of the different observed packets, the metering process function F would choose/filter/aggregate different sets of packets, which would create different flows. In other words, based on various combination of values of {source IP address, destination IP address, TOS}, F(source IP address, destination IP address, TOS) would result in the definition of one or more flows. The function F is referred to as Flow Type.

* Flow Key:
  Each of the fields which belong to
  1. Packet header (e.g. destination IP address)

2. Property of the packet itself (e.g. packet length)
3. Derived from packet treatment (e.g. AS number)

which is used to define a Flow is termed as Flow Key.

* Flow Type:
  A function F which would take input as a set of Flow Keys and produce as output one or more Flows depending on the combination of values for the set of Flow Keys.

* Flow Record:
  A Flow Record contains information about a specific Flow that was observed at an Observation Point. A Flow Record contains measured properties of the Flow (e.g. the total number of bytes of all packets of the Flow) and usually characteristic properties of the Flow (e.g. source IP address).

* Exporting Process:
  The Exporting Process sends Flow Records to one or more devices that collect these (also known as Collecting Processes). The Flow Records are generated by one or more Metering Processes.

* IPFIX Device:
  A device hosting at least an Observation Point, a Metering Process and an Exporting Process. Typically, corresponding Observation Point(s), Metering Process(es) and Exporting Process(es) are co-located at this device, for example at a router.

* Collecting Process:
  The Collecting Process receives Flow Records from one or more Exporting Processes. The Collecting Process might process or store received Flow Record, but such actions are out of the scope of this document.

* Collector:
  The device which hosts one or more Collecting Processes.

* Metering Process:
  The Metering Process generates Flow Records. Input to the process are IP packets observed at an Observation Point and packet treatment at the Observation Point, for example the selected output interface. The Metering Process consists of a set of functions that includes packet header capturing, timestamping, sampling, classifying, and maintaining Flow Records.

* Observation Domain:
  A collection of Observation Points and their corresponding Metering Processes is termed an Observation Domain. The Observation Domain presents a unique ID to the Collecting Process for identifying the export packets generated by it. One or more Observation Domains can interface with the same Exporting Process. Example: The Observation Domain could be a router line-card, composed of several interfaces with each interface being an Observation Point.

* Flow Recording Process:
  The Flows generated from the metering device(s) in an Observation Domain MAY be collected into one or more database before exporting. This functional block in addition to maintaining the Flow database(s) MAY do Flow aggregation, maintain the aggregate statistics etc. This block is optional for an IPFIX device.

* Template:
  Template is an ordered n-tuple (e.g. <type,length>, TLV), used to completely identify the structure and semantics of a particular information that needs to be communicated from the IPFIX Device to the Collector. Each template is uniquely identifiable by some means (e.g. by using a Template ID).

* Control Information, Data Stream:
  The information that needs to be exported from the IPFIX device can be classified into the following categories:

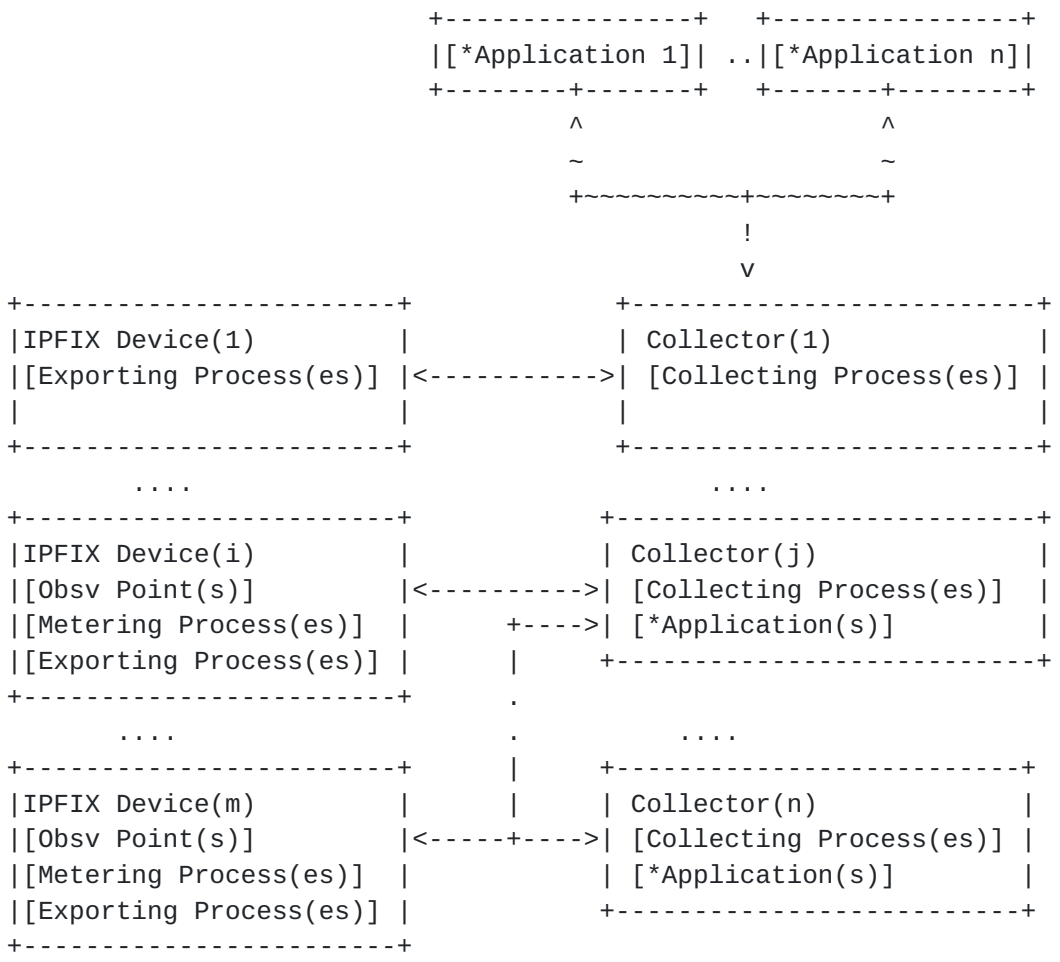
  - Control Information :
    This includes the Flow type definition, selection criteria for packets within the Flow sent by the Exporting Process and any IPFIX protocol messages (e.g. keepalives). The 'control' stream carries all the information needed for the end-points to understand the IPFIX protocol, and specifically for the receiver to understand and interpret the data sent by the sender.

  - Flow Records :
    This includes data records carrying the field values for the various observed Flows at each of the Observation Point. A sequence of such records may also be described as a Data Stream.

The definitions in this section are intended be identical with that of the terminology used in [IPFIX-REQS] with additional terms introduced to help in defining the architecture model.

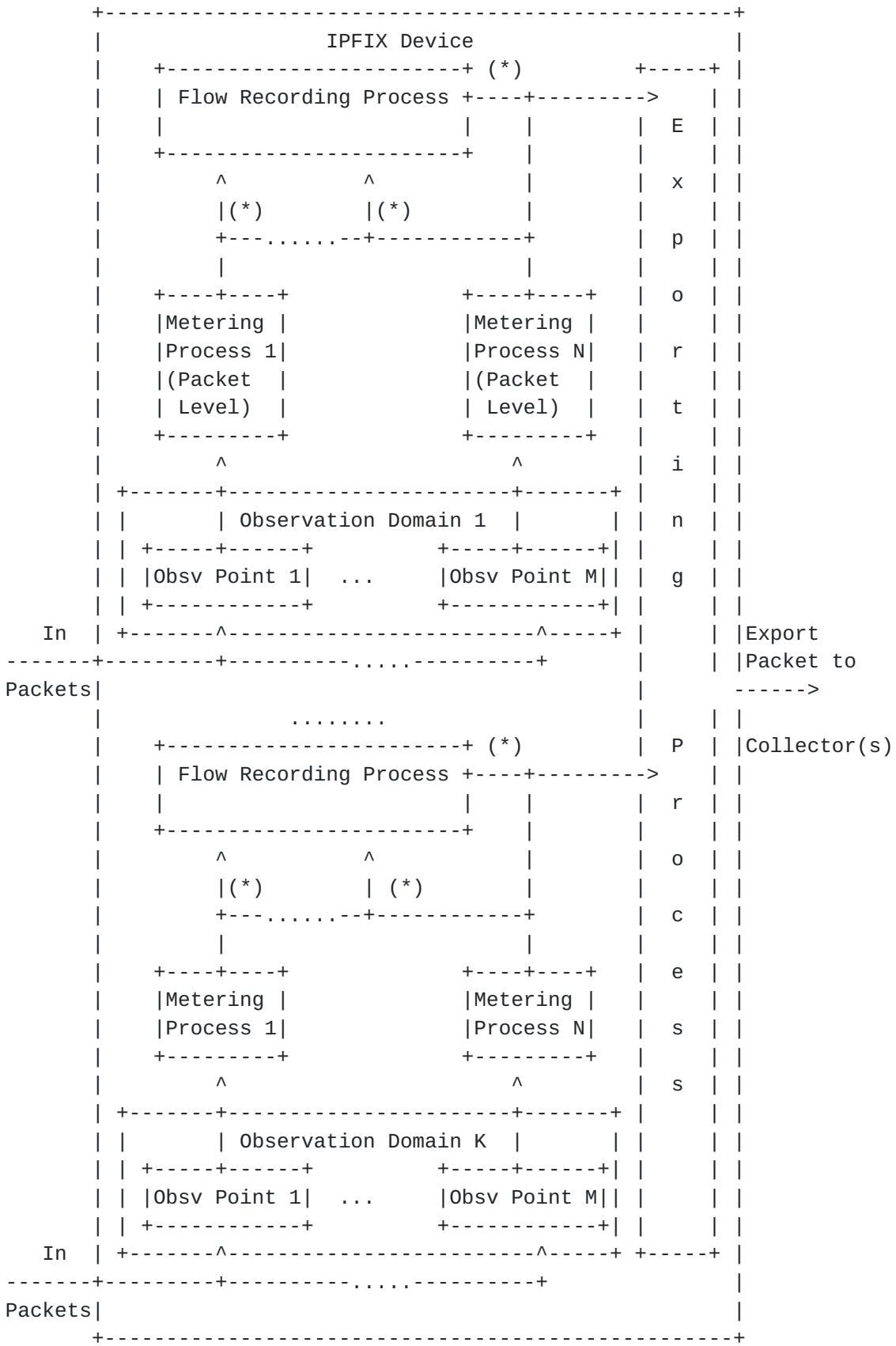
4. IPFIX reference Model

The figure below shows the reference model for IPFIX. This figure covers the various possible scenarios that can exist in an IPFIX system.



The various functional components are indicated within []. The functional components within [*] are not part of the IPFIX framework. The interfaces shown by "<-->" are defined by the IPFIX framework and those shown by "<~~>" are not.

The figure below shows a typical IPFIX device.


IPFIX Device
Flow Recording Process
(*)
Metering Process 1 (Packet Level)
Metering Process N (Packet Level)
Observation Domain 1
Obsv Point 1
Obsv Point M
Observation Domain K
Export Packet to Collector(s)
In Packets

In the above figure the IPFIX components are shown in rectangular boxes. The interface shown by (*) is applicable only if the optional flow recording process is present. Otherwise the metering process(es) at the packet level interface directly with the exporting function. Note that in case of multiple Observation Domains, a unique ID per Observation Pomain must be transmitted as a parameters to the exporting function. The exporting process includes IPFIX protocol and underlying transport layer.

5. IPFIX Functional and Logical blocks

5.1. Metering Process Functions

Every observation point in an IPFIX device, partcipating in flow measurements, MUST be associated with at least one metering process. The packet coming into an observation point goes into each of the metering processes associated with the observation point. Broadly, each metering process extracts the packet headers that come into an observation point, does timestamping and classifies the packet into flow(s) based on the selection criteria.

5.1.1. Flow Classification

The collecting process MUST be able to map the flow record to the corresponding property types defined by the flow type. In addition to flow type, the collector when it receives the flow records, MAY need the following to interpret the flow records further:

a. Observation Point
b. Selection Criteria for Packets

A flow record can be better analyzed if the Observation Point from which it is measured is known. As such it is recommended that the flow record carry the Observation Point information along with the flow records when exported. In cases where there is a single observation point or where the observation point information is not relevant, the metering process MAY choose not to add this to the flow records.

5.1.2. Selection Criteria for Packets

The measurement device MAY define rules so that only certain packets within a flow can be chosen for measurement at an observation point. This MAY be done by one of the two methods defined below or a combination of them. A combination of each of these ways can be

adopted to select the packets, i.e. one can define a set of methods {F1, S1, F2, S2, S3} executed in a certain sequence at an observation point to collect flows of a particular type.

5.1.3. Function on properties that determines a flow type (Fi)

Packets that satisfy a function on the fields defined by the packet header fields or fields obtained while doing the packet processing or the properties of the packet itself.

Example: Mask/Match of the fields that define a filter. The filter may be defined as {Protocol == TCP, Destination Port between 80 and 120}.

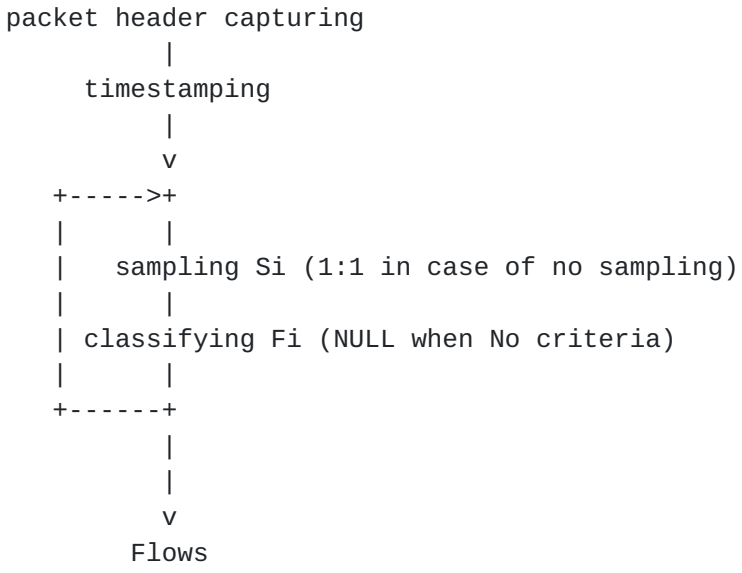
Multiple such filters could be used in any sequence to select packets.

5.1.4. Sampling packets on a flow type (Si)

Packets that satisfy the sampling criteria for this flow type.

Example: Sample every 100th packet that was received at an observation point and collect the flow information for a particular flow type. choosing all the packets is a special case where sampling rate is 1:1.

The figure below shows the operations which MAY be applied as part of a typical metering process.



5.2. Observation Domain

The Observation Domain is a logical block that presents a single identity for a group of Observation Points within an IPFIX device. Each {Observation point, Metering Process} MUST belong to a single Observation Domain. An IPFIX device could have multiple Observation Domains each of which has a subset of the total set of Observation Points in it. Each Observation Domain MUST carry an unique ID within the context of an IPFIX device. One exporting process MAY serve multiple Observation Domains. In such a case the exporting process uses this unique ID to distinguish export packets among the different Observation Domains. The same concept is used at the collecting process also to identify packets from different Observation Domains from the same IPFIX device.

5.3. Flow Recording Process

The Flow Recording Process is a functional block, which manages all the flows generated from an Observation Domain. The typical functions of a Flow Recording Process MAY include:

* Maintain database(s) of all the flows from an Observation Domain. This includes creating new records, updating existing ones, computing flow statistics, deriving further flow properties, adding non-flow specific information (in some cases fields like AS numbers, router state etc.)
* Maintain aggregate statistics like flows generated, flows exported etc.

It is not mandatory that every IPFIX device use a Flow Recording Process. Instead the flows generated by the metering process can be directly sent to the exporting process.

5.4. Exporting Process

The Exporting Process is the functional block that includes one or more instances of IPFIX protocol. On one side it interfaces with metering process/flow recording process to get flow records and on the other side, talks to a collecting processs on the collector(s).

5.5. IPFIX protocol

At the IPFIX device, the protocol functionality resides in the exporting process. The IPFIX protocol gets flows from flow recording process or directly from the metering process, and carries them to the collector(s).

At a high level, the IPFIX protocol at an IPFIX device does the following:

Maintain rules for :

1. Picking and sending control information and flow records.
2. Encoding control and flow record information based on the IPFIX Information Model [IPFIX-IMODEL].
3. Flow expiration.
4. IPFIX device overload handling.
5. Selective export of flow records if any.

Functions :

1. Encode the selected control information into templates.
2. Encode the flows observed at the observation points into flow records.
3. Packetize the selected templates and flow records into IPFIX export packets.
4. Use the underlying transport layer to send the export packets to the collector.
5. Handle export errors and timeouts.
6. Handle IPFIX device overload.
7. Apply selective export filters if any to the flow records.

For details on IPFIX protocol, refer to [IPFIX-PROTO].

6. Encoding Control Information

The following rules provide guidelines to be followed while encoding the control information.

- Per-flow control information SHOULD be encoded such that it can capture the structure and semantics of the corresponding flow data for each of the flows exported by the IPFIX device.
- Configuration control information SHOULD be encoded such that it can capture the structure and semantics of the corresponding configuration data. The configuration data which is also control information, SHOULD carry additional information on the boundary within which the configuration takes effect. Forexample, sampling

using the same sampling algorithm, say 1 in 100 packets is configured on 2 observation points O1 and O2. The configuration in this case MAY be encoded as <ID, boundary (O1,O2), sampling algorithm, interval (1 in 100)> where ID uniquely identifies this configuration.

- There SHOULD be provisions to encode fixed length and variable length fields
- All fields MUST be encoded in network byte order.
- The exporter MUST encode a given field based on the encoding standards prescribed by [IPFIX-PROTO].

7. Encoding Flow Data Information

The following rules provide guidelines to be followed while encoding the flow data information.

- A flow data record SHOULD contain enough information so that the collecting process can identify the corresponding <Per-flow control information, Configuration control information>.
- All fields MUST be encoded in network byte order.
- The exporter MUST encode a given field based on the encoding standards prescribed by [IPFIX-PROTO].

8. Exporting Control Information

The Control Information is used by the collecting process to :

- Decode and interpret flow records.
- Understand the state of the exporting process.

As such sending control information from exporting process in a timely and reliable manner is critical to the proper functionality of the IPFIX collecting process. The following approaches MAY be taken for the export of control information.

1. Send all the control information pertaining to flow records prior to sending the flow records themselves. This includes any incremental changes that happens to the definition of the flow records.

2. Notify on a near real time basis the state of the IPFIX device to the collecting process. This includes all changes such as a configuration change that affects the flow behavior, changes to exporting process resources that alter export rates, etc., which the collector needs to be aware of.

3. Since it is vital that a collecting process maintains accurate knowledge of the exprtoer's state, the export of the control information SHOULD be done such that that it reaches the collector reliably. One way to achieve this would be to send the control information over a reliable transport.

9. Flow Expiration and Export

A flow is considered to be inactive if no packets of this flow have been observed at the observation point for a given timeout interval. The flow MAY be expired and exported under the following conditions:

1. If the Metering Process can deduce the end of a Flow. The Flow SHOULD be exported when the end of the Flow is detected. For example: flow generated by TCP type of traffic where the FIN or RST bits indicate the end of the flow.

2. If the Flow has been inactive for a certain period of time. This inactivity timeout SHOULD be configurable at the Metering Process, with a minimum value of 0 for an immediate expiration. For example: flow generated by UDP type of traffic.

3. For long-running flows, the Exporting Process MAY export the flow records on regular basis. Some of the reasons for doing this would be:

   a. Report the flow records periodic accounting information to the collecting process.
   b. Avoid counter wrapping.

   When a long-running flow is exported, the flow MAY still be maintained in the IPFIX device so that for the incoming packets that continue to come on the same flow, a new flow does not get created in the flow recording data base.

4. In some cases flows MAY be exported as they are generated. This can be useful when real time processing of flow records is required.

5. If the IPFIX device experiences resource constraints, a flow MAY be prematurely expired (example: memory)

6. In some cases flows the exporting process MAY choose not to export the generated flow as is. For example, this happens if the a set of flows are aggregated into more coarse flows.

10. Export Error Handling

This section describes some of the errors that may be encountered:

- In IPFIX protocol while doing export.
- Feedback received by IPFIX protocol from other entities in the export path towards the collector.

1. Unavailability of resources e.g. packet buffers for IPFIX export packets.
2. Error in transport layer.

The protocol MAY choose to do one or more of the following actions:

- Buffer the flow records until the error condition gets corrected.
- Drop flow records for one or more flows based on some rules. In such a case a record of what action is taken MUST be maintained, e.g. n flow records of a flow were dropped.

10.1. Selection Criteria of flows for export

There MAY be additional rules defined within the context observation domain so that only certain flows records are picked up for export. This MAY be done by either one or a combination of Si, Fi which is described in the section on "Selection Criteria for Packets".

Example: Only the flow records which meet the following selection criteria are exported.

1. All flow records whose destination IP address matches {20.3.1.5}.
2. Every other (.i.e. sampling rate 1 in 2) flow record whose destination IP address matches {160.0.1.30}.

11. The Selected IPFIX Protocol

There are existing standard practices in the area of flow export like Netflow, CRANE, LFAP etc. IPFIX's charter mentions to choose the protocol among these existing practices that fits the IPFIX requirements the most. There may be additions or modifications made to the chosen protocol to fit it exactly into the IPFIX architecture.

The working group went through an extensive evaluation of the various existing protocols that are available today weighing the level of compliance with the requirements and architecture and finally selected Netflow V9 with minor modification as the basis for the

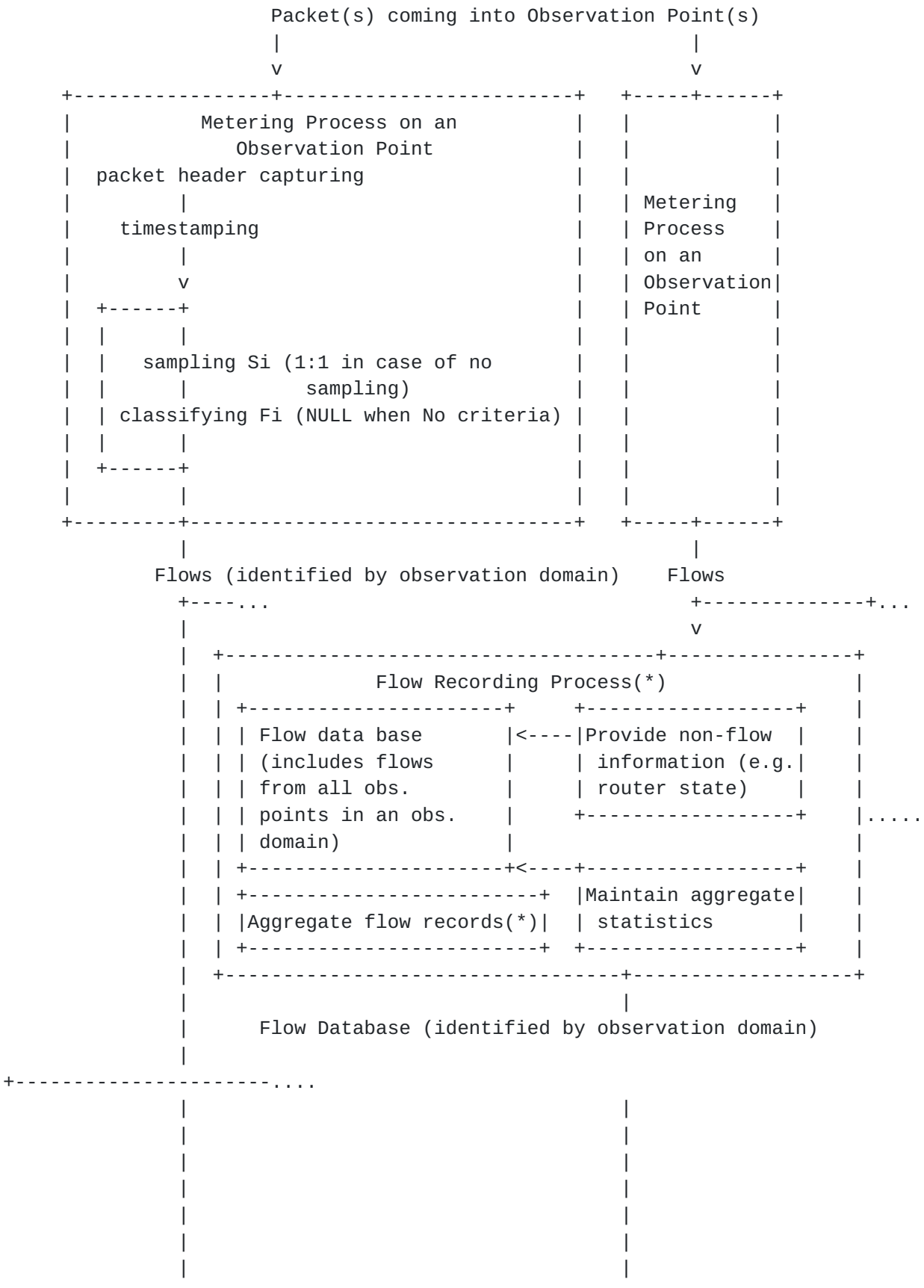
IPFIX protocol. Following is a brief description of the chosen IPFIX basis protocol; details of the IPFIX protocol proper are given in [IPFIX-PROTO].

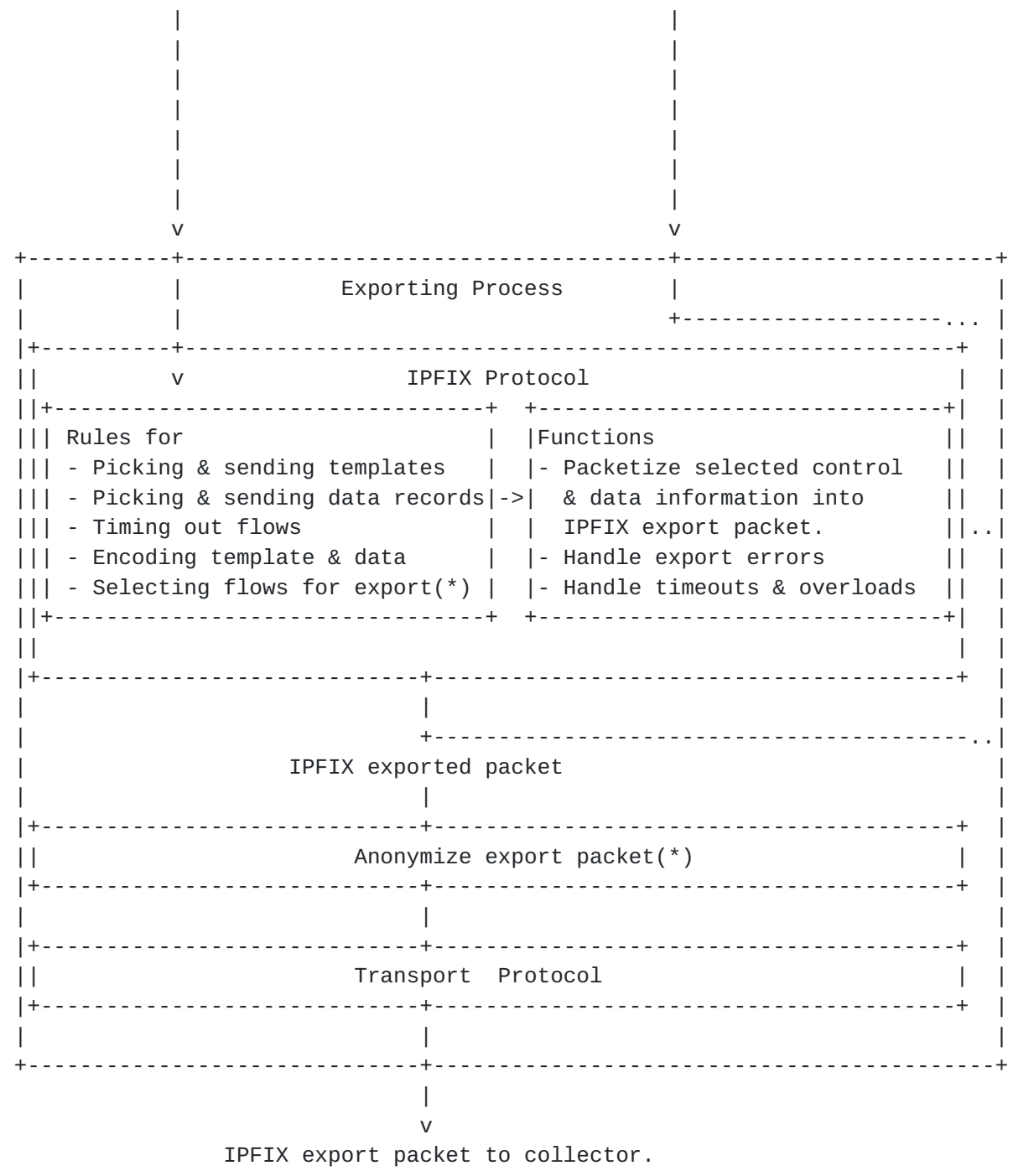
This protocol is template based. A template in terms of Netflow V9 is a collection of fields with corresponding descriptions of their structures and their semantics which is in strict conformance with IPFIX architecture.

This approach provides the following advantages:

- Using the template mechanism, new fields can be added to IPFIX flow records without changing the structure of the export record format.
- Templates that are sent to the collecting process contain the structural information about the exported flow record fields. Therefore, if the collector does not understand the semantics of new fields it can still interpret the flow record.
- Because the template mechanism is flexible, it allows the export of only the required fields from the flows to the collecting process. This helps to reduce the exported flow data volume and possibly provide memory savings at the Exporting Process and Collecting Process. Sending only the required information can also reduce network load.

The figure below shows the functions performed in sequence by the various functional blocks in an IPFIX device.


Packet(s) coming into Observation Point(s)
Metering Process on an Observation Point
packet header capturing
timestamping
sampling Si (1:1 in case of no sampling)
classifying Fi (NULL when No criteria)
Metering Process on an Observation Point
Flows (identified by observation domain)
Flows
Flow Recording Process(*)
Flow data base (includes flows from all obs. points in an obs. domain)
Provide non-flow information (e.g. router state)
Aggregate flow records(*)
Maintain aggregate statistics
Flow Database (identified by observation domain)



(*) indicates that the block is optional.

12. Collecting Process

A Collecting Process is a subsystem that interacts with one or more IPFIX devices. The functions of the collecting process MAY include:

* Identifying, accepting and decoding export packets from different {Exporting Process, Observation Domain} pairs .
* Running the IPFIX protocol.
* Storing the control information and flow records received from IPFIX device.
* Notifying the IPFIX device its status and problems.

At a high level, IPFIX protocol at the collecting process is responsible for the following:

1. Receive and store the control information.
2. Decode and store the flow records using the control information.
3 Optionally monitor the status of the collecting process and execute a fail over in case of problem.

12.1. IPFIX Protocol on Collecting Process

1. Receive and decode the flow records from the IPFIX devices.
2. Ability to indicate flow record losses to the exporting IPFIX device and/or IPFIX users.
3. Optionally notify the status and overload conditions to the IPFIX device.

Once the selection is made from the set of candidate protocols, this section would be replaced by the chosen protocol.

12.2. Support for Applications

Applications that use the information collected by IPFIX may be Billing, Intrusion Detection sub-systems, etc. These applications may be an integral part of the collecting process or collocated with the collecting process. The way by which these applications interface with IPFIX system to get the desired information is out of this document's scope.

12.3. Export Models

12.3.1. Export Model with Reliable Control Connection

As mentioned in the [IPFIX-REQS], the control information and data stream MUST be transported over a congestion-aware transport protocol. If the network in which the IPFIX device and collecting process are located does not guarantee reliability, at least the control information SHOULD be exported over a reliable transport. There could be network security concerns between IPFIX device and collecting process. To avoid re-inventing the wheel, and to reduce the complexity of flow export protocol, one or a combination of the following methods MAY be adopted as a solution to achieve security :

* IP Authentication Header MAY be used when the threat environment requires stronger integrity protections, but does not require confidentiality.
* IP Encapsulating Security Payload (ESP) MAY be used to provide confidentiality and integrity.
* If the transport protocol used is TCP, optionally TCP MD5 signature option MAY be used to protect against spoofed TCP segments.
* If the transport protocol used is TCP, optionally TLS MAY be used to add integrity, authenticity and confidentiality.

The data stream MAY be exported over a reliable or unreliable transport protocol.

As explained above the transport connection (in the case of a connection oriented protocol) is pre-setup between the IPFIX device and the collector. Once connected, the collector side receives the control information and uses this information to interpret the flow records. The IPFIX device SHOULD set the keepalive (e.g. keepalive timeout in the case of TCP; the HEARTBEAT interval in the case of SCTP; IPFIX protocol level Keepalive if any) to a sufficiently low value so that it can quickly detect a collector crash.

12.4. Collector Crash Detection and Recovery

Collector crash refers to crash or restart of collecting process or the collector itself.

12.4.1. Export Model with Reliable Control Connection

The collector crash is detected at the IPFIX device by the break in control connection (depending on the transport protocol the connection timeout mechanisms differ). On detecting a Keepalive timeout, the IPFIX device SHOULD stop sending the flow export data to the collector and try reconnecting the transport connection. This is valid for a single collector scenario. If there are multiple collectors for the same IPFIX device, the IPFIX device opens control connections to each of the collectors. But data gets sent only to one of the collectors which is chosen as the primary. There could be one or more collectors configured as secondary and a priority assigned to them. The primary collector crash is detected at the IPFIX device by the break in control connection (depending on the transport protocol the connection timeout mechanisms differ). On detecting loss of connectivity, the IPFIX device opens data stream with the secondary collector of the next highest priority. This collector now becomes the primary. The maximum export data loss would be the amount of data exported in the time between when the loss of connectivity to the collector happened, and the time at which this was detected by the IPFIX device.

12.5. Collector Redundancy

Since IPFIX protocol requires a congestion-aware transport, achieving redundancy using multicast is not an option. Multiple <control information, data stream> pairs could be setup, each to a different collector from the same IPFIX device. The control and data information are then replicated on each of the control information and data stream. Add text here.

13. IPFIX flow collection from Special Devices

IPFIX could be implemented on devices which perform one or more of the following special services :

* Explicitly drop packets. For example a device which provides firewall service drops packets based on some administrative policy.
* Alter the values of fields used as IPFIX flow keys of interest. For example a device which provides NAT service can change source or(and) destination IP address.

In the cases above, there should be clear guidelines as to

- How and when to classify the packets as flows in the IPFIX device
- What extra information be exported so that the collector can make a clear interpretation of the received flow records.

14. IPFIX flow collection for special traffic

An IPFIX device could be doing one or more of generating, receiving, altering special types of traffic which are listed below.

* Tunnel traffic: The IPFIX device could be the head, midpoint or endpoint of a tunnel. In such cases the IPFIX could be handling GRE, IPinIP, UTI traffic.
* VPN traffic: The IPFIX device could be a Provider Edge device which receives traffic from customer sites belonging to different Virtual Private Networks.

In the cases above, there should be clear guidelines as to

- How and when to classify the packets as flows in the IPFIX device.
- If multiple encapsulations are used to define flows, how to convey the same fields (e.g. IP address) in different layers.
- How to differentiate flows based on different private domains. For example, overlapping IP addresses in Layer-3 VPNs

15. Security Considerations

IP flow information can be used for various purposes, such as usage accounting, traffic profiling, traffic engineering, and intrusion detection. For each application, the security requirement may differ significantly from one to another. To be able to satisfy the security needs of various IPFIX users, the architecture of IPFIX MUST provide different levels of security protection.

15.1. Data security

IPFIX data consists of control information and data stream generated by the IPFIX device.

The IPFIX data may exist in both the IPFIX device and the collector. In addition, the data is also transferred on the wire from the IPFIX device to the collector when it is reported. To provide security, the data SHOULD be protected from adversary.

The protection of IPFIX data within the end system (IPFIX device and collector) is out of the scope. It is assumed that the end system operator will provide adequate security for the IPFIX data.

The IPFIX architecture MUST allow different levels of protection to the IPFIX data on the wire. Where ever security functions are required it is recommended to leverage to lower layers using either IPsec or TLS, if they can successfully satisfy the security requirement of IPFIX data protection.

To protect the data on the wire, three levels of granularity SHOULD be supported:

15.1.1. No security

Security may not be required when the transport between the IPFIX device and the collector is perceived as safe. This option allows the protocol to run most efficiently without extra overhead and an IPFIX solution MUST support it.

15.1.2. Authentication only

The authentication only protection provides the IPFIX users the assurance of data integrity and authenticity. The data exchanged between the IPFIX device and the collector is protected by authentication signature. Any modification of the IPFIX data will be detected by the recipient, resulting in discarding of the received data. However, the authentication only option doesn't offer data confidentiality. The IPFIX user SHOULD avoid use this option when sensitive or confidential information is being exchanged. An IPFIX solution SHOULD support this option. The authentication only option SHOULD provide replay attack protection. Some means to achieve this level of security are:

* TCP with MD5 options.
* IP Authentication Header

15.1.3. Encryption

Data encryption provides the best protection for IPFIX data. The IPFIX data is encrypted at the sender and only the intended recipient can decrypt and have access to the data. This option MUST be used when the transport between the IPFIX device and the collector are unsafe and the IPFIX data needs to be protected. It is recommended to use the underlying security layer functions for this purpose. Some means to achieve this level of security are:

* Encapsulating Security Payload.
* Transport Layer Security Protocol

The data encryption option adds overhead to the IPFIX data transfer. It may limit the rate that an export can report its flow to the collector due to the heavy resource requirement of running encryption.

15.2. IPFIX end point authentication

It is important to make sure that the IPFIX device is talking to the "right" collector instead of a masqueraded collector. The same logic also holds true from the collector point of view that it want to make sure it is collecting the flow information from the "right" IPFIX device. The IPFIX architecture SHOULD allow the authentication capability so that either one-way or mutual authentication can be performed between the IPFIX device and collector.

The IPFIX architecture SHOULD use the existing transport protection protocols such as TLS or IPSEC to fulfill the authentication requirement.

16. IPFIX overload

An IPFIX device could get overloaded under various conditions. This MAY lead to:

- Exhaustion of internal resources used for flow generation and/or export.
-

16.1. Denial of service (DoS) attack prevention

Since one of the potential usages for IPFIX is for intrusion detection, it is important for the IPFIX architecture to support some kind of DoS resistance.

16.1.1. Network under attack

The Network itself may be under attack, resulting in an overwhelming number of IPFIX messages. The IPFIX SHOULD try to capture as much information as possible. However, when large amount IPFIX messages are generated in a short period of time, the IPFIX may become overloaded.

16.1.2. Generic DoS attack on the IPFIX system

The IPFIX system may subject to generic DoS attacks, just as any system on any open networks. These types of attacks are not IPFIX specific. Preventing and responding to such types of attacks are out of the scope of IPFIX WG.

16.1.3. IPFIX Specific DoS attack

There is a specific attack on the IPFIX portion of the IPFIX device or Collector.

- The attacker could pound the Collector with spoofed IPFIX export packets. One way to solve this problem is to periodically synchronize the sequence numbers of the flow records between exporting process and the collecting process.
- The attacker could provide false reports to the IPFIX device by sending spoofed control packets.

The problems mentioned above can be solved to a large extent if the control packets are encrypted both ways.

(To be added and discussed on the general list).

17. IANA Considerations

Need Port number assigned from IANA [more to be written]

18. References

[IPFIX-REQS] J. Quittek ,T. Zseby, B. Claise,"Requirements for IP Flow Information Export", (work in progress) ,Internet Draft, <draft-ietf-ipfix-reqs-11.txt>, June 2003.

[IPFIX-IMODEL] P. Calato, J. Meyer, J. Quittek, "IPFIX: Information Model," work in progress) ,Internet Draft, <draft-ietf-ipfix-info-00.txt>, June 2003.

[IPFIX-PROTO] M. Fulmer, P. Calato, B. Claise, R. Penno, "IPFIX: Protocol," Internet Draft, <draft-ietf-ipfix-protocol-00.txt>, June 2003.

19. Acknowledgements

We wish to thank all the people contributing to the requirements discussion on the mailing list, and the design teams for many valuable comments.

Tanja Zseby
Paul Calato
Dave Plonka
Jeffrey Meyer
Benoit Claise
Ganesh Sadasivan
K.C.Norseth
Vamsi Valluri
Cliff Wang
Ram Gopal
Jc Martin
Carter Bullard
Juergen Quittek
Reinaldo Penno
Nevil Brownlee
Simon Leinen
Kevin Zhang

20. Author's Addresses

Ganesh Sadasivan
Cisco Systems, Inc.
170 W. Tasman Dr.
San Jose, CA 95134
USA
Phone: +1 (408) 527-0251
Email: gsadasiv@cisco.com

Nevil Brownlee
CAIDA | The University of Auckland
Phone: +64 9 373 7599 x8941
E-mail: n.brownlee@auckland.ac.nz

Benoit Claise
Cisco Systems
De Kleetlaan 6a b1
1831 Diegem
Belgium
Phone: +32 2 704 5622
Email: bclaise@cisco.com

Juergen Quittek
NEC Europe Ltd.
Adenauerplatz 6
69115 Heidelberg
Germany
Phone: +49 6221 90511-15
EMail: quittek@ccrle.nec.de

21. Full Copyright Statement