

INTERNET-DRAFT  
Intended Status: Standard Track

Sami Boutros  
Jerome Catrouillet  
VMware

Expires: January 8, 2020

July 7, 2019

MAC move over Geneve encapsulation  
draft-boutros-nvo3-mac-move-over-geneve-01

Abstract

This document specifies a mechanism to signal Media Access Control (MAC) addresses move over a Network Virtualization Overlays over Layer 3 (NVO3) virtual tunnel. Such notification is useful in redundancy scenarios when a Layer 2 service that was active on a Network Virtualization Edge (NVE) fails over to a standby NVE. This notification can be used only when data plane mac learning is enabled over the NVO3 tunnels.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at  
<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at  
<http://www.ietf.org/shadow.html>

Copyright and License Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|     |                                   |   |
|-----|-----------------------------------|---|
| 1   | Introduction . . . . .            | 3 |
| 1.1 | Terminology . . . . .             | 3 |
| 1.2 | Abbreviations . . . . .           | 3 |
| 2.  | MAC Move Frame Format . . . . .   | 5 |
| 3.  | Operation . . . . .               | 6 |
| 3.1 | Operation of Sender . . . . .     | 6 |
| 3.2 | Operation of Receiver . . . . .   | 7 |
| 4.  | Security Considerations . . . . . | 7 |
| 5.  | IANA Considerations . . . . .     | 7 |
| 6   | References . . . . .              | 8 |
| 6.1 | Normative References . . . . .    | 8 |
| 6.2 | Informative References . . . . .  | 8 |
|     | Authors' Addresses . . . . .      | 8 |

## 1 Introduction

In multi-homing scenarios a Layer 2 service can be multi-homed to more than one Network virtualization Edge (NVE). Only one NVE can be active for a given Layer 2 service, and a standby NVE can be chosen to take over the Layer 2 service when the active NVE goes down. The mechanisms to elect which NVE will be active or standby to provide single active redundancy for a given Layer 2 service is outside the scope of this document.

When a standby NVE gets activated, Standby NVE needs to send a MAC Move message to all remote NVE(s) that spans this L2 service over the Geneve tunnels to Move all MAC learned in data plane via the old active NVE.

The MAC Move message will contain the NVE Identifier(s) of the old Active NVE and the new active NVE.

MAC Move can be used to optimize network convergence and reduce blackholes, when an active NVE hosting a logical L2 service fails over to a standby NVE.

The protocol defined in this document addresses possible loss of the MAC Move messages due to network congestion, but does not guarantee delivery.

In the event that MAC Move messages does not reach the intended target, the fallback to MAC re-learning or as a last resort aging out of MAC addresses in the absence of frames from the sources, will resume the traffic via new active NVE.

### 1.1 Terminology

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 [RFC2119].

### 1.2 Abbreviations

NVO3 Network Virtualization Overlays over Layer 3

OAM Operations, Administration, and Maintenance

TLV Type, Length, and Value

VNI Virtual Network Identifier

NVE Network Virtualization Edge

NVA Network Virtualization Authority

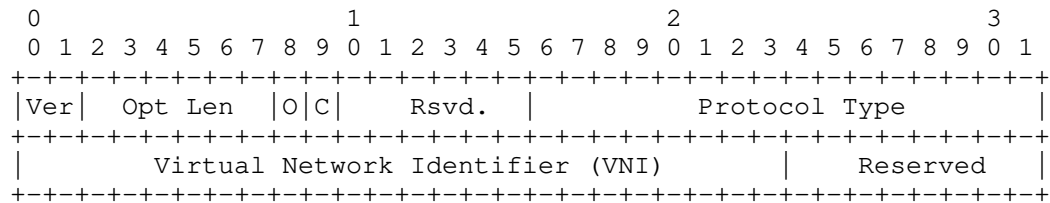
NIC Network interface card

VTEP Virtual Tunnel End Point

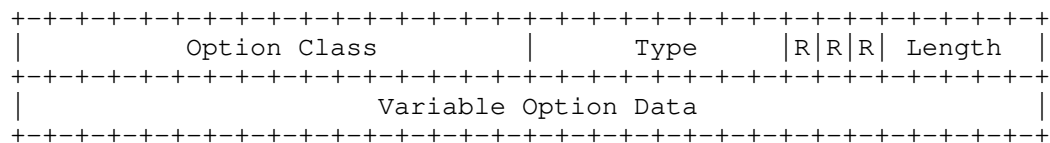
Transit device Underlay network devices between NVE(s).

## 2. MAC Move Frame Format

Geneve Header:



Geneve Option Header:



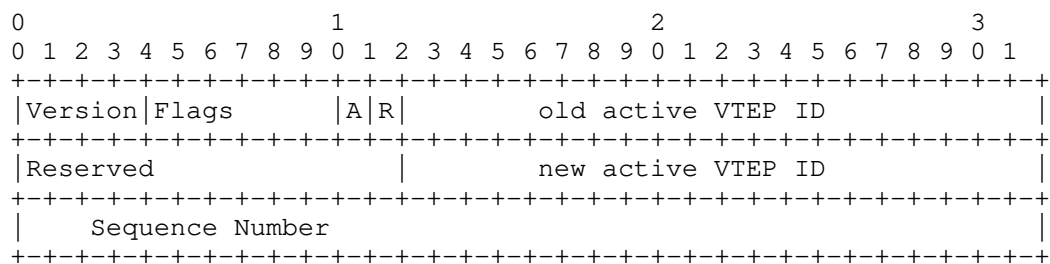
Option Class = To be assigned by IANA (TBA).

Type = TBA.

'C' bit, Endpoints must drop if they do not recognize this option)

Length = 2 (8 bytes)

Variable option data:



Version (4 bits): Initially the Version will be 0.

A (1 bit): Set by a receiver to acknowledge receipt and processing of a MAC Move message.

R (1 bit): Set to indicate if the sender is requesting reset of the sequence numbers. The sender sets this bit when it has no local record of previous send and expected receive sequence numbers.

Flags(6): Reserved and should be set to 0.

VTEP ID (20 bits): Identifies old and new active NVE(s).

Sequence Number (32) bits: For overflow detection a sequence number that exceeds (0x7FFFFFFF) is considered an overflow and reset to 1.

### 3. Operation

This section describes how the initial MAC Move Messages are sent and retransmitted, as well as how the messages are processed and retransmitted messages are identified. The mechanisms described are very similar to the one defined in [RFC 7769].

#### 3.1 Operation of Sender

At the NVE , each L2 logical switch identified by a VNI is associated with a counter to keep track of the sequence number of the transmitted MAC Move messages. Whenever a node sends a MAC Move message, it increments the transmitted sequence-number counter and includes the new sequence number in the message.

The transmit sequence number is initialized to 1 at the onset, after the wrap and after the sequence number reset request receipt. Hence the transmit sequence number is set to 2 in the first MAC Move message sent after the sequence number is initialized to 1.

The sender expects an ACK from the receiver within a retransmit time interval, which can be either a default (1 second) or a configured value. If the ACK is not received within the Retransmit time, the sender retransmits the message with the same sequence number as the original message. The retransmission MUST cease when an ACK is received. In order to avoid continuous re-transmissions in the absence of acknowledgements, the sender MUST cease retransmission after a small number of transmissions, two retries is RECOMMENDED.

Alternatively, an increasing backoff delay with a larger number of retries MAY be implemented to improve scaling issues.

During the period of retransmission, if a need to send a new MAC Move message with updated sequence number arises, then retransmission of the older unacknowledged Move message MUST be suspended and

retransmit time for the new sequence number MUST be initiated. In essence, a sender engages in retransmission logic only for the most recently sent Move message for a given L2 Logical Switch identified by a VNI.

In the event that the L2 logical switch is deleted and re-added or the VTEP node is restarted with new configuration, the NVE may lose information about the previously sent sequence number. This becomes problematic for the remote peer as it will continue to ignore the received MAC Move messages with lower sequence numbers. In such cases, it is desirable to reset the sequence numbers, the reset R-bit is set in the first MAC Move to notify the remote peer to reset the send and receive sequence numbers. The R-bit must be cleared in subsequent MAC Move messages after the acknowledgement is received.

### 3.2 Operation of Receiver

Each L2 logical switch identified by a VNI is associated with a receive sequence number per remote NVE to keep track of the expected sequence number of the MAC Move message.

Whenever a MAC Move message is received, and if the sequence number on the message is greater than the value in the receive sequence number of this remote NVE, the MAC addresses learned from the NVE associated with the NVE identifier in the message are moved to be associated with the new active NVE identifier, and the receive sequence number of the remote NVE is updated with the received sequence number. The receiver sends an ACK with the same sequence number in the received message.

If the sequence number in the received message is smaller than or equal to the value in the receive sequence number per remote NVE, the MAC Move is not processed. However, an ACK with the received sequence number MUST be sent as a response to stop the sender retransmission.

A MAC Move message with the R-bit set MUST be processed by resetting the receive sequence number of the remote NVE, and Moving the MACs as described above. The acknowledgement is sent with the R-bit cleared.

### 4. Security Considerations

This document does not introduce any additional security constraints.

### 5. IANA Considerations

IANA is requested to assign a new option class from the "Geneve Option Class" registry for the Geneve MAC Move option.

| Option Class | Description |
|--------------|-------------|
|--------------|-------------|

-----  
XXXX

-----  
Geneve MAC Move

## 6 References

### 6.1 Normative References

[KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

### 6.2 Informative References

[Geneve] "Generic Network Virtualization Encapsulation", [I-D.ietf-nvo3-geneve]  
[RFC 7769] "MAC Address Withdrawal over Static PW", [RFC 7769]

## Authors' Addresses

Sami Boutros  
VMware  
Email: [boutross@vmware.com](mailto:boutross@vmware.com)

Jerome Catrouillet  
VMware  
Email: [jcatrouillet@vmware.com](mailto:jcatrouillet@vmware.com)

Ankur Sharma  
VMware  
Email: [ankursharma@vmware.com](mailto:ankursharma@vmware.com)



INTERNET-DRAFT  
Intended Status: Standard Track

Sami Boutros  
Jerome Catrouillet  
Sri Mohana Singamsetty  
VMware  
Parag Jain  
Cisco

Expires: March 22, 2020

September 19, 2019

MAC move over Geneve encapsulation  
draft-boutros-nvo3-mac-move-over-geneve-02

## Abstract

This document specifies a mechanism to signal Media Access Control (MAC) addresses move over a Network Virtualization Overlays over Layer 3 (NVO3) virtual tunnel. Such notification is useful in redundancy scenarios when a Layer 2 service that was active on a Network Virtualization Edge (NVE) fails over to a standby NVE. This notification can be used only when data plane mac learning is enabled over the NVO3 tunnels.

## Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at  
<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at  
<http://www.ietf.org/shadow.html>

## Copyright and License Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|     |                                   |   |
|-----|-----------------------------------|---|
| 1   | Introduction . . . . .            | 3 |
| 1.1 | Terminology . . . . .             | 3 |
| 1.2 | Abbreviations . . . . .           | 3 |
| 2   | MAC Move Frame Format . . . . .   | 5 |
| 3   | Operation . . . . .               | 6 |
| 3.1 | Operation of Sender . . . . .     | 6 |
| 3.2 | Operation of Receiver . . . . .   | 7 |
| 4   | Security Considerations . . . . . | 7 |
| 5   | IANA Considerations . . . . .     | 7 |
| 6   | References . . . . .              | 8 |
| 6.1 | Normative References . . . . .    | 8 |
| 6.2 | Informative References . . . . .  | 8 |
|     | Authors' Addresses . . . . .      | 8 |

## 1 Introduction

In multi-homing scenarios a Layer 2 service can be multi-homed to more than one Network virtualization Edge (NVE). Only one NVE can be active for a given Layer 2 service, and a standby NVE can be chosen to take over the Layer 2 service when the active NVE goes down. The mechanisms to elect which NVE will be active or standby to provide single active redundancy for a given Layer 2 service is outside the scope of this document.

When a standby NVE gets activated, Standby NVE needs to send a MAC Move message to all remote NVE(s) that spans this L2 service over the Geneve tunnels to Move all MAC learned in data plane via the old active NVE.

The MAC Move message will contain the NVE Identifier(s) of the old Active NVE and the new active NVE.

MAC Move can be used to optimize network convergence and reduce blackholes, when an active NVE hosting a logical L2 service fails over to a standby NVE.

The protocol defined in this document addresses possible loss of the MAC Move messages due to network congestion, but does not guarantee delivery.

In the event that MAC Move messages does not reach the intended target, the fallback to MAC re-learning or as a last resort aging out of MAC addresses in the absence of frames from the sources, will resume the traffic via new active NVE.

### 1.1 Terminology

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 [RFC2119].

### 1.2 Abbreviations

NVO3 Network Virtualization Overlays over Layer 3

OAM Operations, Administration, and Maintenance

TLV Type, Length, and Value

VNI Virtual Network Identifier

NVE Network Virtualization Edge

NVA Network Virtualization Authority

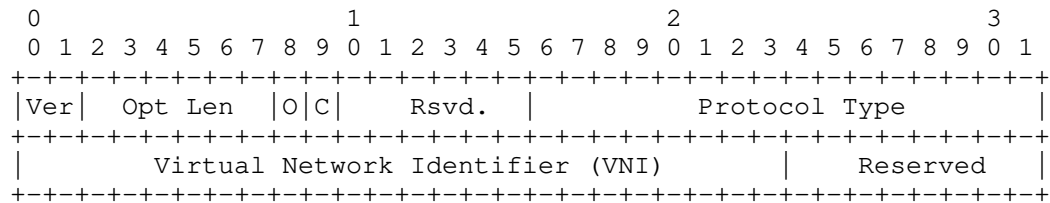
NIC Network interface card

VTEP Virtual Tunnel End Point

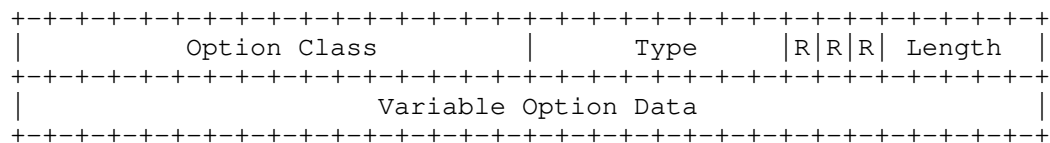
Transit device Underlay network devices between NVE(s).

## 2. MAC Move Frame Format

Geneve Header:



Geneve Option Header:



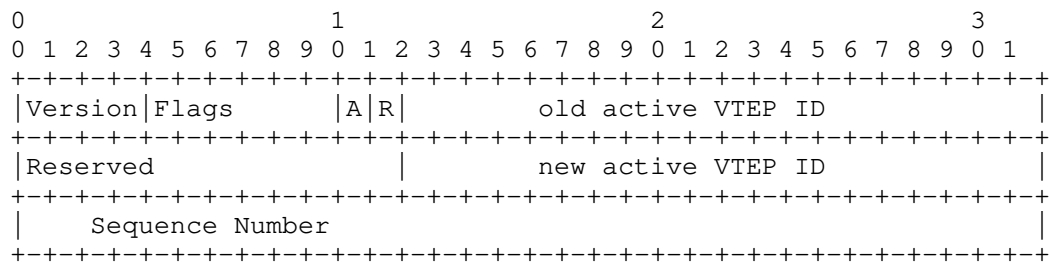
Option Class = To be assigned by IANA (TBA).

Type = TBA.

'C' bit, Endpoints must drop if they do not recognize this option)

Length = 2 (8 bytes)

Variable option data:



Version (4 bits): Initially the Version will be 0.

A (1 bit): Set by a receiver to acknowledge receipt and processing of a MAC Move message.

R (1 bit): Set to indicate if the sender is requesting reset of the sequence numbers. The sender sets this bit when it has no local record of previous send and expected receive sequence numbers.

Flags(6): Reserved and should be set to 0.

VTEP ID (20 bits): Identifies old and new active NVE(s).

Sequence Number (32) bits: For overflow detection a sequence number that exceeds (0x7FFFFFFF) is considered an overflow and reset to 1.

### 3. Operation

This section describes how the initial MAC Move Messages are sent and retransmitted, as well as how the messages are processed and retransmitted messages are identified. The mechanisms described are very similar to the one defined in [RFC 7769].

#### 3.1 Operation of Sender

At the NVE , each L2 logical switch identified by a VNI is associated with a counter to keep track of the sequence number of the transmitted MAC Move messages. Whenever a node sends a MAC Move message, it increments the transmitted sequence-number counter and includes the new sequence number in the message.

The transmit sequence number is initialized to 1 at the onset, after the wrap and after the sequence number reset request receipt. Hence the transmit sequence number is set to 2 in the first MAC Move message sent after the sequence number is initialized to 1.

The sender expects an ACK from the receiver within a retransmit time interval, which can be either a default (1 second) or a configured value. If the ACK is not received within the Retransmit time, the sender retransmits the message with the same sequence number as the original message. The retransmission MUST cease when an ACK is received. In order to avoid continuous re-transmissions in the absence of acknowledgements, the sender MUST cease retransmission after a small number of transmissions, two retries is RECOMMENDED.

Alternatively, an increasing backoff delay with a larger number of retries MAY be implemented to improve scaling issues.

During the period of retransmission, if a need to send a new MAC Move message with updated sequence number arises, then retransmission of

the older unacknowledged Move message MUST be suspended and retransmit time for the new sequence number MUST be initiated. In essence, a sender engages in retransmission logic only for the most recently sent Move message for a given L2 Logical Switch identified by a VNI.

In the event that the L2 logical switch is deleted and re-added or the VTEP node is restarted with new configuration, the NVE may lose information about the previously sent sequence number. This becomes problematic for the remote peer as it will continue to ignore the received MAC Move messages with lower sequence numbers. In such cases, it is desirable to reset the sequence numbers, the reset R-bit is set in the first MAC Move to notify the remote peer to reset the send and receive sequence numbers. The R-bit must be cleared in subsequent MAC Move messages after the acknowledgement is received.

### 3.2 Operation of Receiver

Each L2 logical switch identified by a VNI is associated with a receive sequence number per remote NVE to keep track of the expected sequence number of the MAC Move message.

Whenever a MAC Move message is received, and if the sequence number on the message is greater than the value in the receive sequence number of this remote NVE, the MAC addresses learned from the NVE associated with the NVE identifier in the message are moved to be associated with the new active NVE identifier, and the receive sequence number of the remote NVE is updated with the received sequence number. The receiver sends an ACK with the same sequence number in the received message.

If the sequence number in the received message is smaller than or equal to the value in the receive sequence number per remote NVE, the MAC Move is not processed. However, an ACK with the received sequence number MUST be sent as a response to stop the sender retransmission.

A MAC Move message with the R-bit set MUST be processed by resetting the receive sequence number of the remote NVE, and Moving the MACs as described above. The acknowledgement is sent with the R-bit cleared.

### 4. Security Considerations

This document does not introduce any additional security constraints.

### 5. IANA Considerations

IANA is requested to assign a new option class from the "Geneve Option Class" registry for the Geneve MAC Move option.

| Option Class | Description     |
|--------------|-----------------|
| -----        | -----           |
| XXXX         | Geneve MAC Move |

## 6 References

### 6.1 Normative References

[KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

### 6.2 Informative References

[Geneve] "Generic Network Virtualization Encapsulation", [I-D.ietf-nvo3-geneve]  
[RFC 7769] "MAC Address Withdrawal over Static PW", [RFC 7769]

## Authors' Addresses

Sami Boutros  
VMware  
Email: boutross@vmware.com

Jerome Catrouillet  
VMware  
Email: jcatrouillet@vmware.com

Sri Mohana Singamsetty  
Email: msingamsetty@vmware.com

Ankur Sharma  
VMware  
Email: ankursharma@vmware.com

Parag Jain  
Cisco  
Email: paragj@cisco.com



NVO3  
Internet-Draft  
Intended status: Standards Track  
Expires: December 29, 2017

D. Migault  
June 27, 2017

Geneve Security Architecture  
draft-mglt-nvo3-geneve-security-architecture-00

Abstract

This document describes the Geneve Security Architecture.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 29, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|   |    |
|---|----|
| 1. Requirements notation . . . . .                    | 2  |
| 2. Introduction . . . . .                             | 2  |
| 3. Terminology . . . . .                              | 3  |
| 4. Architecture Overview . . . . .                    | 4  |
| 5. Geneve Security Policies Database . . . . .        | 5  |
| 5.1. Selectors . . . . .                              | 6  |
| 5.2. Geneve Security Policies . . . . .               | 8  |
| 5.3. Geneve Security Policies Example . . . . .       | 8  |
| 6. Geneve Security Association Database . . . . .     | 11 |
| 6.1. Geneve Security Associations . . . . .           | 11 |
| 7. Geneve Security Module Packet Processing . . . . . | 13 |
| 7.1. Outbound Geneve Processing . . . . .             | 13 |
| 7.2. Inbound Geneve Packet Processing . . . . .       | 13 |
| 8. IANA Considerations . . . . .                      | 14 |
| 9. Security Considerations . . . . .                  | 14 |
| 10. Acknowledgment . . . . .                          | 14 |
| 11. References . . . . .                              | 15 |
| 11.1. Normative References . . . . .                  | 15 |
| 11.2. Informational References . . . . .              | 15 |
| Author's Address . . . . .                            | 16 |

## 1. Requirements notation

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 [RFC2119].

## 2. Introduction

[I-D.ietf-nvo3-encap] and [I-D.mglt-nvo3-security-requirements] clearly state the need to secure the Geneve overlay network and provide means to authenticate the Geneve Header as well as being able to encrypt the Geneve Payload.

Both of these requirements are fulfilled with specific Geneve Security Options. More explicitly, [I-D.mglt-nvo3-geneve-authentication-option] defines an option that authenticate the Geneve fixed Header and optionally a set of Geneve Option as well as a portion of the Geneve Payload. [I-D.mglt-nvo3-geneve-encryption-option] defines a Geneve Option that enables to encrypt a subset of Geneve Options as well as a portion of the Geneve Payload. Further descriptions on how an Geneve Security Option is treated is out of the scope of this document.

This document defines how the Geneve overlay can be secured properly. A Geneve Element may handle different Geneve overlay networks

associated with different level of security. This document defines how to associate a level of security to an Geneve overlay network. In addition, a security level for a given overlay network may result in a combination of multiple Geneve Security Options. As the order these Geneve Security Options are processed matters, it is necessary the sending and receiving Geneve Element have similar behaviours in order to guarantee interoperability while securing a Geneve overlay Network.

This document explains how Geneve Security Policies and Geneve Security Associations are organized to associate a given level of security to an Geneve overlay network. In addition, this document also exposes how the Geneve Security Module implementing the security interacts with the Geneve architecture.

### 3. Terminology

- o Geneve Elements: designates all elements that handled Geneve Packets. These elements may be terminal elements such as NVEs for example, but can also be on path elements that are expected to manage the flow inside the Geneve overlay network.
- o Geneve Packet: designates the packet that Geneve Elements are expected to handled. It is composed of a Geneve Header and a Geneve Payload. In this document the Outer Ethernet Header and Outer IPv4 Header as well as the Outer UDP Header defined in [I-D.ietf-nvo3-geneve] section 3.1 are not part of the Geneve Packet. Similarly, the Outer Ethernet Header, the Outer IPv6 Header as well as the Outer UDP Header defined in [I-D.ietf-nvo3-geneve] section 3.2 are not part of the Geneve Packet.
- o Geneve Header: is described in [I-D.ietf-nvo3-geneve] section 3.4. The Geneve Header may contain zero or more Geneve options.
- o Geneve Payload: designates the data carried by a Geneve Packet. [I-D.ietf-nvo3-geneve]. In [I-D.ietf-nvo3-geneve] section 3.1 and section 3.2, the Geneve Payload would be the Inner Ethernet Header, the Payload and the Frame Check Sequence.
- o Geneve Fix Header: The Geneve Header without any Geneve Options.
- o Geneve Security Policies (GSP):
- o Geneve Security Policies Data Base (GSP DB):
- o Geneve Security Association (GSA):

- o Geneve Security Association Data Base (GSA DB):
- o Geneve Security Options (GSO): A security option defined for Geneve. Currently the security options that have been defined are GAO or GEO.
- o Geneve Authentication Option (GAO): Geneve Option that describes how to authenticate the Geneve Header as well as part of the Geneve Payload. This option is described in [I-D.mglt-nvo3-geneve-authentication-option].
- o Geneve Encryption Option (GEO): Geneve Option that describe how to encrypt Geneve Options as well as a part of the Geneve Payload. GEO is defined in [I-D.mglt-nvo3-geneve-encryption-option].
- o Geneve Security Module: an implementation responsible to enforce the security of Geneve Packets.

#### 4. Architecture Overview

The Geneve Security Architecture is represented in figure Figure 1. Geneve security is enforced by the Geneve Security Module. The Geneve Security Policies (GSP) define which flows inside a virtual network needs to be secured by associating a action SECURE, BYPASS or DISCARD to each Geneve Packet. When a Geneve Packet is tagged as SECURE, the GSP provides specific structures known as Geneve Security Associations (GSA) that describe how the Geneve Packet MUST be secured. Typically, the GSA defines the type of option (Geneve Authentication Option (GAO) or the Geneve Encryption Option (GEO) to be computed or validated, as well as the necessary material such as the appropriated counters, the necessary keys to compute and validate the GSO. GSP and GSA are respectively stored in GSP Database (GSP DB) and GSA Database (GSA DB).

For outbound traffic, the Geneve Security Module receives a non secured Geneve Packet and is responsible to secure that Geneve Packet with the appropriated GSOs - as defined by the GSP/GSAs. Once the GSO have been added, Outer Encapsulation is performed as described in [I-D.ietf-nvo3-geneve] i.e. the Geneve Packet is being encapsulated with Outer Ethernet / Outer IPv4 or IPv6 / and Outer UDP.

For inbound traffic, the Geneve Security Module defines whether a incoming Geneve Packet must be secured or not as defined by the GSP. If a Geneve Packet does not have to be secured, any GSO found is ignored. Otherwise, the Geneve Security Module validates each GSO, and check the validated GSOs are conformed to the defined GSP. The last step is necessary to make sure that in addition to valid security options, the expected GSO were encountered.

This document assumes that all nodes GSP DB and GSA DB are appropriately provisioned by the control plane.

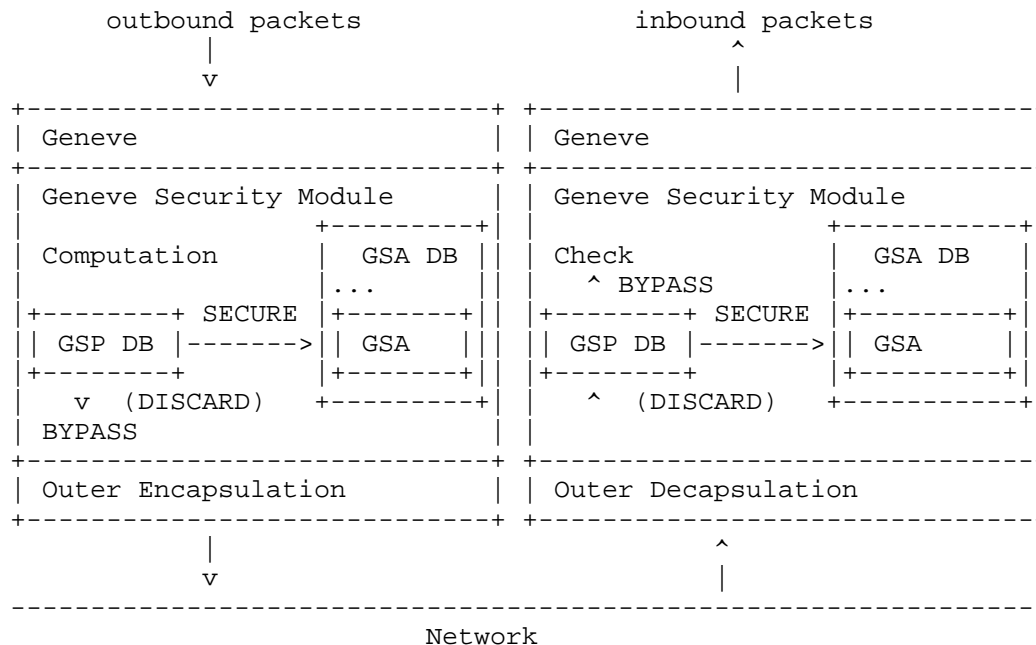


Figure 1: Geneve Security Architecture

## 5. Geneve Security Policies Database

The GSP DB contains a list of GSP that associates a Geneve Packet with a specific action `BYPASS`, `DISCARD`, `SECURE`.

The matching between a Geneve Packet and an action is performed through Selectors. These Selectors associated to specific values defined whether a Geneve Packet match a given GSP. As GSO may result in encrypting a Selector, a GSP lookup is always performed with a "clear text" Geneve Packet. More specifically, the GSP lookup for a Geneve Packet associated with the SECURE action is performed before the GSO is being added or after the GSO has been validated. For outbound Geneve Packet, a GSP DB look up is performed using the Selectors' value before the GSO is computed. In that case, the GSP will even provide the required structure to generate the GSO. On the other hand, for incoming traffic, the GSO is identified by an identifier carried by the Geneve Packet, a GSP look up is performed once the GSO has been validated / decrypted. As a consequence, the same GSP DB is shared by the sending and the receiving Geneve Element.

When BYPASS is selected, then the Geneve Security Module forwards the matching Geneve to the next layer. As represented in Figure 1, the next layer of an outbound Geneve Packet is the Outer Encapsulation while the next layer of an incoming Geneve Packet is the Geneve layer. When DISCARD is selected, the Geneve Security Module is expected to drop the matching Geneve Packet. When a SECURE action is selected the associated GSAs MUST be provided. For outbound Geneve Packet, the GSAs provided will be used in order to appropriately generate the GSO. On the other hand, for incoming Geneve Packet, the GSAs are returned so the Geneve Security Module can validate the GSO present in the Geneve Packet are conform to the GSP.

It is worth noting that for incoming Geneve Packet, those not tagged as BYPASS or DISCARD are by default considered as tagged as SECURE. This means that the GSP DB may be split into sub databases that contains GSP associated to a specific action. GSP DB (DISCARD/BYPASS) may contain all GSP associated to the DISCARD and BYPASS rules while GSP DB (SECURE) contains all GSP associated to the action SECURE. By doing so, a incoming Geneve Packet may be associated to the SECURE action without performing a lookup on GSP DB (SECURE). This does not prevents the Geneve Security Module from validating the GSO found in the incoming Geneve Packet, as these GSO are carrying a specific Identifier. On the other hand, the GSP DB (SECURE) MUST be lookup in order to validate that all GSO defined by the security policies have been appropriately validated.

### 5.1. Selectors

The Selectors are the elements read from the packet in order to match a GSP. When a Selector is expected to be found in the Geneve Packet, the Selectors values that match the condition are indicated with a range or a list of matching values. For clarity, this document uses ANY to indicate the full range. When the Selector may not exist or may not be accessible and must be ignored to evaluate the matching condition, it is qualified of being OPAQUE.

Geneve Header Selectors:

- o Geneve Version (2 bits): The version of the Geneve Version. This field is always present and MUST be specified. When all Geneve Version are associated to the same GSP, then all values must be specified with ANY = [0, ..., 4].
- o OAM bit (1 bit): The indication of an OAM indication. When OAM and non OAM traffic is associated to the same GPS, then all values must be indicated with ANY = [0, 1].

- o Critical bit ( 1 bit): indicates the presence of a critical option. When the presence of a critical option or its absence are associated to the same GSP, then all values must be indicated with ANY = [0, 1]
- o Rsv (6 bits): Currently [I-D.ietf-nvo3-geneve] specifies the field is set to zero by the sender and ignored by the receiver. According to these rules, the sender is expected to DISCARD any non zero values and the receiver is expect to indicate all these values in its GSP.
- o Protocol Type (16 bits): indicates the type of the Geneve Payload. It is likely that only a few types will be specified for matching.
- o VNI: indicates the virtual network identifier. It is also likely that only a small set of VNI values will be provisioned per switches.
- o Reserved (8 bits): (see Rsv)
- o Geneve Options Class - Type List: This fields specifies the Geneve Options that MUST be present. The absence of one of these option result in discarding the Geneve Header. When the presence or absence of a specific Geneve Option has no impact for the GSP selection, the value is set to OPAQUE as they may not be any options.

Additional Selectors are considered within the Geneve Payload. The Selectors provided below are expected to enable different GSP according to the protection of the traffic. Typically, the Geneve overlay network may protect differently traffic that is already protected by the tenants with IPsec, DTLS/TLS, or SSH.

- o Next Header (IPv6) / Protocol (IPv4) (8 bits): For IPv6 this field indicates the presence of an IPv6 Option or the transport layer used after the IPv6 Header. For IPv4 packets, the protocol indicates the layer after the IPv4 Header. This field is typically used to indicate whether IPsec/AH (51), IPsec/ESP (50), TCP (6) or UDP (17) is used. Next Header is a mandatory field and is expected in any IP header. When the matching condition does not consider the Next Header or Protocol number than ANY = [0, ..., 65535] is expected. When non IP packet are expected, OPAQUE is expected.
- o Port Source is typically used to determine how the tenants traffic is being protected by TLS or DTLS. Ports associated to TLS are expected to be 443 https, 636 ldaps, 989 ftps-data, 990 ftps, 992 telnets, 993 imaps, 994 ircs, 995 pop3s, 5061 sips, 22 ssh/scp.

Note that not all transport are associated with a port number. When only transport layers with port numbers are expected to be used (such as TCP or UDP) and the matching condition does not consider the port numbers, ANY = [0, ..., 65535] is expected. When traffic may not have port numbers - such as ICMP traffic, OPAQUE is expected.

- o Port Destination: (see Port Source)

## 5.2. Geneve Security Policies

Geneve Security Policies are unidirectional. A GSP is composed of:

- o Selectors that express a matching condition
- o Action that defines if the Geneve Packet MUST be DISCARDED, BYPASSED or SECURED. When the associated action is SECURE, then the GSP associates an ordered list of GSA. The GSA contains the description and the necessary material to perform the SECURE action.

The GSP DB is an ordered list of GSP.

## 5.3. Geneve Security Policies Example

According to [I-D.ietf-nvo3-geneve], the associated Geneve Version is 0, a sender MUST set Rsv and Reserved to zero. When the sender only supports [I-D.ietf-nvo3-geneve], it may performed a sanity check for its outbound packets. The rules can be places at the beginning of the GSP DB.



| Selector         | Value                    | Action  |
|------------------|--------------------------|---------|
| Geneve Version   | [1 ... 4] (non-zero)     | DISCARD |
| OAM              | [0,1] (ANY)              |         |
| Critical         | [0,1] (ANY)              |         |
| Rsv              | [0, ..., 63] (ANY)       |         |
| Protocol         | [0, ..., 65535] (ANY)    |         |
| VNI              | [0, ..., 16777215] (ANY) |         |
| Reserved         | [0, ..., 255] (ANY)      |         |
| Geneve Options   | OPAQUE                   |         |
| Next Header      | [0, ..., 255] (ANY)      |         |
| Port Source      | OPAQUE                   |         |
| Port Destination | OPAQUE                   |         |
| Geneve Version   | [0 ... 4] (ANY)          | DISCARD |
| OAM              | [0,1] (ANY)              |         |
| Critical         | [0,1] (ANY)              |         |
| Rsv              | [1, ..., 63] (non-zero)  |         |
| VNI              | [0, ..., 65535] (ANY)    |         |
| Reserved         | [0, ..., 15] (ANY)       |         |
| Geneve Options   | OPAQUE                   |         |
| Next Header      | [0, ..., 255] (ANY)      |         |
| Port Source      | OPAQUE                   |         |
| Port Destination | OPAQUE                   |         |
| Geneve Version   | [0 ... 4] (ANY)          | DISCARD |
| OAM              | [0,1] (ANY)              |         |
| Critical         | [0,1] (ANY)              |         |
| Rsv              | [1, ..., 63] (ANY)       |         |
| VNI              | [0, ..., 65535] (ANY)    |         |
| Reserved         | [1, ..., 15] (non-zero)  |         |
| Geneve Options   | OPAQUE                   |         |
| Next Header      | [0, ..., 255] (ANY)      |         |
| Port Source      | OPAQUE                   |         |
| Port Destination | OPAQUE                   |         |

Figure 2: Example 1: Geneve Security Policy for [I-D.ietf-nvo3-geneve] compliance (sender)

By default a Geneve Security Module may DISCARD any Geneve packet that have no matching This is indicated by the following GSP at the end of the GSP DB.

| Selector         | Value                    | Action  |
|------------------|--------------------------|---------|
| Geneve Version   | [0 ... 4] (ANY)          | DISCARD |
| OAM              | [0,1] (ANY)              |         |
| Critical         | [0,1] (ANY)              |         |
| Rsv              | [0, ..., 63] (ANY)       |         |
| Protocol         | [0, ..., 65535] (ANY)    |         |
| VNI              | [0, ..., 16777215] (ANY) |         |
| Reserved         | [0, ..., 255] (ANY)      |         |
| Geneve Options   | OPAQUE                   |         |
| Next Header      | [0, ..., 255] (ANY)      |         |
| Port Source      | OPAQUE                   |         |
| Port Destination | OPAQUE                   |         |

Figure 3: Example 2: Geneve Security Policy for [I-D.ietf-nvo3-geneve] compliance (sender)

The example below details a GSP that proceeds to a specific treatment to the traffic between tenant using ESP. The specific treatment could typically only authenticate the Geneve Packet or partially encrypt the Geneve Payload, in order to only hide the Inner headers - including the ESP header - up to the ESP payload.

In the example, the GSP apply the same GSAs whatever the Geneve Header informations are. More specifically, all virtualized network share the same GSAs.

| Selector         | Value                    | Action                 |
|------------------|--------------------------|------------------------|
| Geneve Version   | [0 ... 4] (ANY)          | SECURE<br>[GSA1, GSA2] |
| OAM              | [0,1] (ANY)              |                        |
| Critical         | [0,1] (ANY)              |                        |
| Rsv              | [0, ..., 63] (ANY)       |                        |
| Protocol         | [0, ..., 65535] (ANY)    |                        |
| VNI              | [0, ..., 16777215] (ANY) |                        |
| Reserved         | [0, ..., 255] (ANY)      |                        |
| Geneve Options   | OPAQUE                   |                        |
| Next Header      | [50] (ESP)               |                        |
| Port Source      | OPAQUE                   |                        |
| Port Destination | OPAQUE                   |                        |

Figure 4: Example 3: Geneve Security Policy for ESP protect traffic

## 6. Geneve Security Association Database

GSA DB contains all GSAs. GSA are expected to contain all the necessary information for the Geneve Security Module to compute the GSO by both the sender and the receiver. This includes for example the cryptographic keys to encrypt (resp. authenticate) as well as to decrypt (resp. validate) the Geneve Packet. In addition, the GSA also contains parameters associated to the protection of the security option such as the anti-replay mechanisms as well as the management of that options such as its lifetime.

For outbound traffic, the concerned GSA are provided by the GSP. In this case, it is the purpose of the implementation of Geneve Security Module to provide that appropriated reference. Most likely, the appropriated GSAs will be designated using a memory address.

For inbound traffic, the concerned GSA is designated by the associated GSO with a GSO-ID. In that case the appropriated GSA is retrieved using this index.

### 6.1. Geneve Security Associations

A GSA contains the following information:

- o GSO ID: The identifier of that GSA. This identifier is used by receiver to bind the appropriated GSO to the appropriated GSA. Note that when the packet is encrypted by the GSO, it may not be possible to associate the GSA using GSP.
- o GSO Protocol: The security protocol associated with the Geneve Security Option. Currently the two GSO are GAO and GEO.

When the security option includes some encryption operation, the following parameters are provided. Note that as recommended by [I-D.ietf-ipsecme-rfc7321bis], encryption is authenticated encryption.

- o GSO Encryption Algorithm: In most cases, the encryption is combined with an authentication performed with the same key.
- o GSO Encryption Key:

When the security option includes a dedicated authentication operation (that is not part of the encryption), the following parameters are provided:

- o GSO Authentication Algorithm:

- o GSO Authentication Key:

The following parameters indicate the coverage of the security

- o GSO Payload Covered Length: the length of the Geneve Payload covered by the GSO. The expression of the length can be a number of bytes, but it may also be defined with an abstract designation. For example, a sending node may be willing to authenticate the Geneve Payload up to the ESP layer. In that case, the sending node will have to compute the corresponding Payload Covered Length. This value is only used by the sending node. The receiving node read that value from the GAO.
- o GSO Covered Geneve Options: Indicates the Geneve Options covered by the GSO. This indication is primarily necessary for the sending node and is derived from the Geneve Packet by the receiving node. It might be checked by the receiving node to validate the GSA. It might typically be expressed as a list of Geneve Options that needs to be covered by the authentication.

In order to implement the anti replay mechanisms the following parameters are provided:

- o GSO Sequence Number Size: indicates the size of the SN. This document considers a 32 bit or a 64 bit length.
- o GSO Last Received Packet: that designates the Sequence Number last sent or received packet.
- o GSO Anti Replay Window: that indicates the minimum acceptable value of the Sequence Number. Any Geneve Packet with a lower SN MUST be rejected. Such SN value is usually derived from the Last Received Packet - Anti Replay Windows.

In order to check the conformity with the GSP:

- o GSO Selectors: The selectors are provided so the receiver can check the Geneve Packet protected by the GSO is conform to the GSP. In other words a valid GSO is not sufficient for the Geneve Packet to be forwarded to the upper layers. Note that the Selectors MUST match the Geneve Packet associated to the GSA before the GSO is built for outbound Packets. For inbound Geneve Packet the Selectors are those that corresponds to the Geneve Packet after the GSO has been validated/decrypted. Selectors are mostly expected to be used by the GSA for incoming Geneve Packet, in order to check the GSA is conform with its GSP.
- o GSA Life time:

## 7. Geneve Security Module Packet Processing

This section assumes that the GSA is valid. Unvalid GSA MUST be deleted or considered as non existing by either the sender or the receiver.

### 7.1. Outbound Geneve Processing

- o The Geneve Security Module consults the GSP DB to determine the GSP associated to the Geneve Packet.
  - \* When a Geneve Packet is DISCARD, the Geneve Packet is dropped.
  - \* When a Geneve Packet is BYPASS, the Geneve Packet is directly forwarded to the lower layers for the outer encapsulation.
  - \* When a Geneve Packet is SECURE, the GSP returns one or multiple Geneve Security Association (GAS) of the Geneve Security Association Database (GSA DB). GAS contains the necessary material to compute the GSO for outbound Geneve Packet. When multiple GAS are returned, GAS are applied in the order they are provided. Each computed GSO carries a unique GSA-ID, so the receiver can check the corresponding GSO without performing a GSP DB lookup.
  - \* When no matching is found, the Geneve Packet is DISCARDED
- o Geneve Packet is forwarded to the lower layers for the Outer Encapsulation.

### 7.2. Inbound Geneve Packet Processing

For inbound Geneve Packets:

- o The Geneve Security Module checks the Geneve Packet is associated to a DISCARD or a BYPASS GSP.
  - \* If a match occurs the Geneve Packet is either DISCARDED or BYPASSED to the Geneve layer.
  - \* Otherwise the Geneve Packet is expected to be SECURED and processed as such by the Geneve Security Module.

When the Geneve Packet is believed to be SECURED.

- o The Geneve Security Module opens a security context which lists the encountered and validated GSO as well as their respective order.

- o The Geneve Security Module inspects the Geneve Header for GSO in a network order and proceeds as follows:
  - \* The Geneve Security Module extracts the GSA-ID of the GSO.
  - \* The Geneve Security Module performs a GSA DB lookup based on the GSA-ID to retrieve the GSA associated to the Geneve Packet.
    - + If the GSA DB the GSO, the SO is skipped and the Geneve Security Module continue wity the next GSO. In this case, the GSO is treated as a unexpected geneve option.
  - \* The Geneve Security Module validates the Geneve Security Option against the GSA. If the validation does not succeeds, the Geneve Packet is discarded.
  - \* The Geneve security Module validates the Geneve Packet - once the GSO process has been performed - is conformed with the GSP by checking the resulting Geneve Packet matches the Selectors provided in the GSA.
    - + If the validation is successful, the Geneve Security Module associates the GSA-ID with a validated status in the security context. For this reason it is important to match the GSA Selector with the appropriated Selectors value. In case multiple GSO are combined, the Selectors of the GSA MAY differ from those used for the GSP DB matching.
    - + If a mismatch occurs the Geneve Packet is dropped.

When all Geneve Security Options have been validated, the Geneve Packet is matched against the GSP DB to validate the GSA-ID listed in the security context match those returned by the GSP DB. Note that the receiver and the sender MUST have the same GSA-IDs, however, computation and validation are processed in a different order.

## 8. IANA Considerations

There are no IANA consideration for this document.

## 9. Security Considerations

## 10. Acknowledgment

## 11. References

### 11.1. Normative References

- [I-D.ietf-ipsecme-rfc7321bis]  
Wouters, P., Migault, D., Mattsson, J., Nir, Y., and T. Kivinen, "Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH)", draft-ietf-ipsecme-rfc7321bis-06 (work in progress), June 2017.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.

### 11.2. Informational References

- [I-D.ietf-nvo3-encap]  
Boutros, S., Ganga, I., Garg, P., Manur, R., Mizrahi, T., Mozes, D., and E. Nordmark, "NVO3 Encapsulation Considerations", draft-ietf-nvo3-encap-00 (work in progress), June 2017.
- [I-D.ietf-nvo3-geneve]  
Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-04 (work in progress), March 2017.
- [I-D.mglt-nvo3-geneve-authentication-option]  
Migault, D., "Geneve Authentication Option", July 2017, <<https://tools.ietf.org/html/I-D.ietf-nvo3-geneve-authentication-option-00>>.
- [I-D.mglt-nvo3-geneve-encryption-option]  
Migault, D., "Geneve Encryption Option", July 2017, <<https://tools.ietf.org/html/I-D.ietf-nvo3-geneve-encryption-option-00>>.
- [I-D.mglt-nvo3-security-requirements]  
Migault, D., "Geneve Security Requirements", July 2017, <<https://tools.ietf.org/html/I-D.mglt-nvo3-security-requirements-00>>.

Author's Address

Daniel Migault

Email: [daniel.migault@ericsson.com](mailto:daniel.migault@ericsson.com)



NVO3  
Internet-Draft  
Intended status: Informational  
Expires: September 1, 2019

D. Migault  
Ericsson  
S. Boutros  
D. Wings  
VMware, Inc.  
S. Krishnan  
Kaloom  
February 28, 2019

Geneve Security Requirements  
draft-mglt-nvo3-geneve-security-requirements-06

Abstract

The document defines the security requirements to protect tenants overlay traffic against security threats from the NVO3 network components that are interconnected with tunnels implemented using Generic Network Virtualization Encapsulation (Geneve).

The document provides two sets of security requirements: 1. requirements to evaluate the data plane security of a given deployment of Geneve overlay. Such requirements are intended to evaluate a given deployment. 2. requirement a security mechanism need to fulfill to secure any deployment of Geneve overlay deployment

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 1, 2019.

## Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|   |    |
|---|----|
| 1. Requirements Notation . . . . .                    | 3  |
| 2. Introduction . . . . .                             | 3  |
| 3. Terminology . . . . .                              | 6  |
| 4. Security Threats . . . . .                         | 6  |
| 4.1. Passive Attacks . . . . .                        | 6  |
| 4.2. Active Attacks . . . . .                         | 7  |
| 5. Requirements for Security Mitigations . . . . .    | 8  |
| 5.1. Protection Against Traffic Sniffing . . . . .    | 8  |
| 5.1.1. Operational Security Requirements . . . . .    | 9  |
| 5.1.2. Geneve Security Requirements . . . . .         | 10 |
| 5.2. Protecting Against Traffic Injection . . . . .   | 10 |
| 5.2.1. Operational Security Requirements . . . . .    | 12 |
| 5.2.2. Geneve Security Requirements . . . . .         | 13 |
| 5.3. Protecting Against Traffic Redirection . . . . . | 13 |
| 5.4. Protecting Against Traffic Replay . . . . .      | 14 |
| 5.4.1. Geneve Security Requirements . . . . .         | 14 |
| 5.4.2. Geneve Security Requirements . . . . .         | 15 |
| 5.5. Security Management . . . . .                    | 15 |
| 5.5.1. Operational Security Requirements . . . . .    | 15 |
| 5.5.2. Geneve Security Requirements . . . . .         | 15 |
| 6. IANA Considerations . . . . .                      | 16 |
| 7. Security Considerations . . . . .                  | 16 |
| 8. Appendix . . . . .                                 | 16 |
| 8.1. DTLS . . . . .                                   | 16 |
| 8.1.1. Operational Security Requirements . . . . .    | 16 |
| 8.1.2. Geneve Security Requirements . . . . .         | 18 |
| 8.2. IPsec . . . . .                                  | 20 |
| 8.2.1. Operational Security Requirements . . . . .    | 20 |
| 8.2.2. Geneve Security Requirements . . . . .         | 21 |
| 9. Acknowledgments . . . . .                          | 23 |
| 10. References . . . . .                              | 23 |

|  |    |
|--|----|
| 10.1. Normative References . . . . .   | 23 |
| 10.2. Informative References . . . . . | 25 |
| Authors' Addresses . . . . .           | 25 |

## 1. Requirements Notation

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

## 2. Introduction

The network virtualization overlay over Layer 3 (NVO3) as depicted in Figure 1, allows an overlay cloud provider to provide a logical L2/L3 interconnect for the Tenant Systems TSes that belong to a specific tenant network. A packet received from a TS is encapsulated by the ingress Network Virtualization Edge (NVE). The encapsulated packet is then sent to the remote NVE through a tunnel. When reaching the egress NVE of the tunnel, the packet is decapsulated and forwarded to the target TS. The L2/L3 address mappings to the remote NVE(s) are distributed to the NVEs by a logically centralized Network Virtualization Authority (NVA) or using a distributed control plane such as Ethernet-VPN. In a datacenter, the NVO3 tunnels can be implemented using Generic Network Virtualization Encapsulation (Geneve) [I-D.ietf-nvo3-geneve]. Such Geneve tunnels establish NVE-to-NVE communications, may transit within the data center via Transit device. The Geneve tunnels overlay network enable multiple Virtual Networks to coexist over a shared underlay infrastructure, and a Virtual Network may span a single data center or multiple data centers.

The underlay infrastructure on which the multi-tenancy overlay networks are hosted, can be owned and provided by an underlay provider who may be different from the overlay cloud provider.

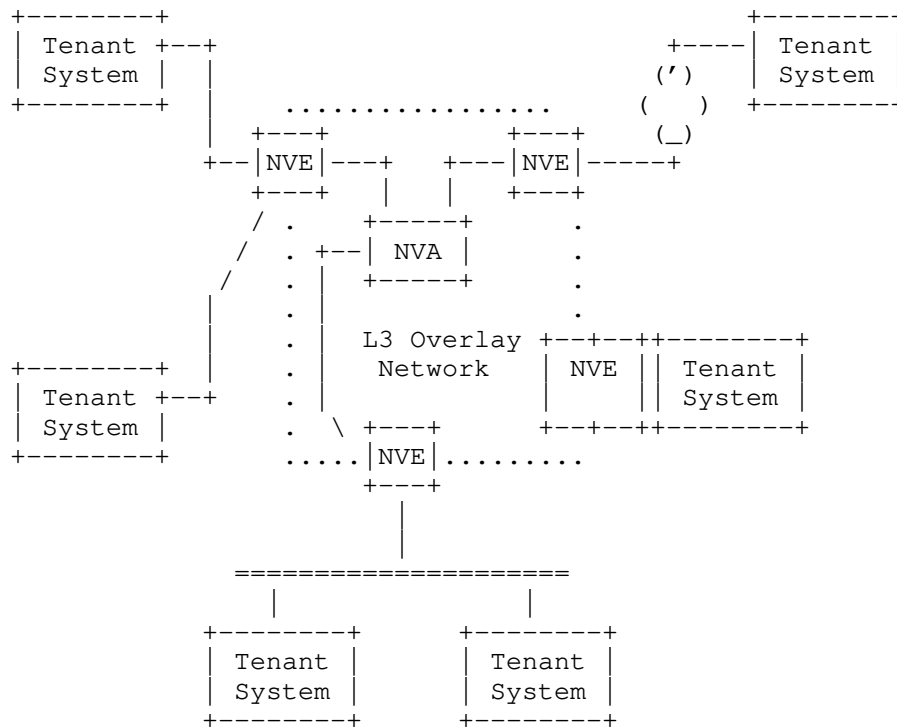


Figure 1: Generic Reference Model for Network Virtualization Overlays [RFC7365]

This document discusses the security risks that a Geneve based NVO3 network may encounter. In addition, this document lists the requirements to protect the Geneve packet components defined in [I-D.ietf-nvo3-geneve] that include the Geneve tunnel IP and UDP header, the Geneve Header, Geneve options, and inner payload.

The document provides two sets of security requirements:

1. SEC-OP: requirements to evaluate a given deployment of Geneve overlay. Such requirements are intended to Geneve overlay provider to evaluate a given deployment. Security of the Geneve packet may be achieved using various mechanisms. Typically, some deployments may use a limited subset of the capabilities provided by Geneve and rely on specific assumptions. Given these specificities, the secure deployment of a given Geneve deployment may be achieved reusing specific mechanisms such as for example DTLS [RFC6347] or IPsec [RFC4301]. On the other hand, the definition of a security mechanisms that enables to secure any Geneve deployment requires the design of a Geneve specific

mechanism. Note that the security is limited to the security of the data plane only. Additional requirements for the control plane MAY be considered in [I-D.ietf-nvo3-security-requirements]. A given Geneve deployment will be considered secured when matching with all SEC-OP requirements does not raise any concern. As such the given deployment will be considered passing SEC-OP requirements that are not applicable.

2. SEC-GEN: requirements a security mechanism need to fulfill to secure any deployment of Geneve overlay deployment. Such mechanism may require the design of a specific solution. In the case new protocol needs to be design, the document strongly recommend to re-use existing security protocols like IP Security (IPsec) [RFC4301] and Datagram Transport Layer Security (DTLS) [RFC6347], and existing encryption algorithms (such as [RFC8221]), and authentication protocols. A given candidate for a security mechanism will be considered as valid when matching with all SEC-GEN requirements does not raise any concern. In other words, at least all MUST status are met.

This document assumes the following roles are involved: - Tenant: designates the entity that connects various systems within a single virtualized network. The various system can typically be containers, VMs implementing a single or various functions.

- Geneve Overlay Provider: provides the Geneve overlay that seamlessly connect the various Tenant Systems over a given virtualized network.

- Infrastructure Provider: provides the infrastructure that runs the Geneve overlay network as well as the Tenant System. A given deployment may consider different infrastructure provider with different level of trust. Typically the Geneve overlay network may use a public cloud to extend the resource of a private cloud. Similarly, a edge computing may extend its resources using resource of the core network.

Tenant, Geneve Overlay Provider and Infrastructure Provider can be implemented by a single or various different entities with different level of trust between each other. The simplest deployment may consists in a single entity running its systems in its data center and using Geneve in order to manage its internal resources. A more complex use case may consider that a Tenant subscribe to the Geneve Overlay Provider which manage the virtualized network over various type of infrastructure. The trust between the Tenant, Geneve Overlay Provider and Infrastructure Provider may be limited.

Given the different relations between Tenant, Geneve Overlay Provider and Infrastructure Provider, this document aims providing requirements to ensure: 1. The Geneve Overlay Provider delivers

tenant payload traffic (Geneve inner payload) and ensuring privacy and integrity. 2. The Geneve Overlay Provider provides the necessary means to prevent injection or redirection of the Tenant traffic from a rogue node in the Geneve overlay network or a rogue node from the infrastructure. 3. The Geneve Overlay Provider can rely on the Geneve overlay in term of robustness and reliability of the signaling associated to the Geneve packets (Geneve tunnel header, Geneve header and Geneve options) in order to appropriately manage its overlay.

### 3. Terminology

This document uses the terminology of [RFC8014], [RFC7365] and [I-D.ietf-nvo3-geneve].

### 4. Security Threats

This section considers attacks performed by NVE, network devices or any other devices using Geneve, that is when the attackers knowing the details of the Geneve packets can perform their attacks by changing fields in the Geneve tunnel header, base header, Geneve options and Geneve inner payload. Attacks related to the control plane are outside the scope of this document. The reader is encouraged to read [I-D.ietf-nvo3-security-requirements] for a similar threat analysis of NVO3 overlay networks.

Threats include traffic analysis, sniffing, injection, redirection, and replay. Based on these threats, this document enumerates the security requirements.

Threats are divided into two categories: passive attack and active attack.

Threats are always associated with risks and the evaluation of these risks depend among other things on the environment.

#### 4.1. Passive Attacks

Passive attacks include traffic analysis (noticing which workloads are communicating with which other workloads, how much traffic, and when those communications occur) and sniffing (examining traffic for useful information such as personally-identifiable information or protocol information (e.g., TLS certificate, overlay routing protocols)).

Passive attacks may also consist in inferring information about a virtualized network or some Tenant System from observing the Geneve traffic. This could also involve the correlation between observed

traffic and additional information. For example, a passive network observer can determine two virtual machines are communicating by manipulating activity or network activity of other virtual machines on that same host. For example, the attacker could control (or be otherwise aware of) network activity of the other VMs running on the same host, and deduce other network activity is due to a victim VM.

A rogue element of the overlay Geneve network under the control of an attacker may leak and redirect the traffic from a virtual network to the attacker for passive monitoring [RFC7258].

Avoiding leaking information is hard to enforced. The security requirements provided in section {{sniffing}} expect to mitigate such attacks by lowering the consequences, typically making leaked data unusable to an attacker.

#### 4.2. Active Attacks

Active attacks involve modifying Geneve packets, injecting Geneve packets, or interfering with Geneve packet delivery (such as by corrupting packet checksum). Active attack may target the Tenant System or the Geneve overlay.

There are multiple motivations to inject illegitimate traffic into a tenants network. When the rogue element is on the path of the TS traffic, it may be able to inject and receive the corresponding messages back. On the other hand, if the attacker is not on the path of the TS traffic it may be limited to only inject traffic to a TS without receiving any response back. When rogue element have access to the traffic in both directions, the possibilities are only limited by the capabilities of the other on path elements - Transit device, NVE or TS - to detect and protect against the illegitimate traffic. On the other hand, when the rogue element is not on path, the surface for such attacks remains still quite large. For example, an attacker may target a specific TS or application by crafting a specific packet that can either generate load on the system or crash the system or application. TCP syn flood typically overload the TS while not requiring the ability to receive responses. Note that udp application are privileged target as they do not require the establishment of a session and are expected to treat any incoming packets.

Traffic injection may also be used to flood the virtual network to disrupt the communications between the TS or to introduce additional cost for the tenant, for example when pricing considers the traffic inside the virtual network. The two latest attacks may also take advantage of applications with a large factor of amplification for their responses as well as applications that upon receiving a packet

interact with multiple TS. Similarly, applications running on top of UDP are privileged targets.

Note also that an attacker that is not able to receive the response traffic, may use other channels to evaluate or measure the impact of the attack. Typically, in the case of a service, the attacker may have access, for example, to a user interface that provides indication on the level of disruption and the success of an attack, Such feed backs may also be used by the attacker to discover or scan the network.

Preventing traffic to cross virtual networks, reduce the surface of attack, but rogue element main still perform attacks within a given virtual network by replaying a legitimate packet. Some variant of such attack also includes modification of unprotected parts when available in order for example to increase the payload size.

## 5. Requirements for Security Mitigations

The document assumes that Security protocols, algorithms, and implementations provide the security properties for which they are designed, an attack caused by a weakness in a cryptographic algorithm is out of scope. The algorithm used MUST follow the cryptographic guidance such as [RFC8247], [RFC8221] or [RFC7525]. In this context, when the document mentions encryption, it assumes authenticated encryption.

Protecting network connecting TSeS and NVEs which could be accessible to outside attackers is out of scope.

An attacker controlling an underlying network device may break the communication of the overlays by discarding or delaying the delivery of the packets passing through it. The security consideration to prevent this type of attack is out of scope of this document.

Securing communication between NVAs and NVEs is out of scope.

Selectively providing integrity / authentication, confidentiality / encryption of only portions of the Geneve packet is in scope. This will be the case if the Tenant Systems uses security protocol to protect its communications.

### 5.1. Protection Against Traffic Sniffing

The inner payload, unless protection is provided by the Tenant System reveals the content of the communication. This may be mitigate by the Tenant using application level security such as, for example JSON Web Encryption [RFC7516] or transport layer security such as DTLS



[RFC6347] or TLS [RFC8446] or IPsec/ESP [RFC4303]. However none of these security protocols are sufficient to protect the entire inner payload. IPsec/ESP still leave in clear the optional L2 layer information as well as the IP addresses and some IP options. In addition to these pieces of information, the use of TLS or DTLS reveals the transport layer protocol as well as ports. As a result, the confidentiality protection of the inner packet may be handled either entirely by the Geneve Overlay Provider, or partially by the Tenant or handled by both the Tenant and the Geneve Overlay Provider.

The Geneve Header contains information related to the Geneve communications or metadata designated as Geneve Information. Geneve Information is carried on the Geneve Outer Header, the Geneve Header (excluding Geneve Options) as well as in the Geneve Options. Geneve Information needs to be accessed solely by a NVE or transit device while other Geneve Information may need to be accessed by other transit devices. More specifically, a subset of the information contained in the Geneve Header (excluding Geneve Options) as well as a subset of (none, one or multiple Geneve Option) may be accessed by a transit device or the NVE while the others needs to be accessed by other transit devices. The confidentiality protection of the Geneve Information is handled by the Geneve Overlay Provider.

In addition to Geneve Information, the traffic generated for the Geneve overlay may be exposed to traffic volumetry and pattern analysis within a virtualized network. Confidentiality protection against traffic pattern recognition is handled by the Geneve Overlay Provider.

#### 5.1.1.1. Operational Security Requirements

A secure deployment of a Geneve overlay must fulfill the requirement below:

- o SEC-OP-1: A secure deployment of a Geneve overlay SHOULD by default encrypt the inner payload. A Geneve overlay provider MAY disable this capability for example when encryption is performed by the Tenant System and that level of confidentiality is believed to be sufficient. In order to provide additional protection to traffic already encrypted by the Tenant the Geneve network operator MAY partially encrypt the clear part of the inner payload.
- o SEC-OP-2: A secure deployment of a Geneve overlay MUST evaluate the information associated to the leakage of Geneve Information carried by the Geneve Packet. When a risk analysis concludes that the risk of leaking sensitive information is too high, such Geneve Information MUST NOT be transmit in clear text.

- o SEC-OP-3: A secure deployment of a Geneve overlay MUST evaluate the risk associated to traffic pattern recognition. When a risk has been identified, traffic pattern recognition MUST be addressed with padding policies as well as generation of dummy packets.

#### 5.1.2. Geneve Security Requirements

A Geneve security mechanism must fulfill the requirements below:

- o SEC-GEN-1: Geneve security mechanism MUST provide the capability to encrypt the inner payload.
- o SEC-GEN-2: Geneve security mechanism SHOULD provide the capability to partially encrypt the inner payload header.
- o SEC-GEN-3: Geneve security mechanism MUST provide means to encrypt a single or a set of zero, one or multiple Geneve Options while leave other Geneve Options in clear. Reversely, a Geneve security mechanism MUST be able to leave a Geneve option in clear, while encrypting the others.
- o SEC-GEN-4: Geneve security mechanism MUST provide means to encrypt the information of Geneve Header (excluding Geneve Options). Reversely, a Geneve security mechanism MUST be able to leave in clear Geneve Header information (Geneve Options excluded) while encrypting the other.
- o SEC-GEN-5: Geneve security mechanisms MUST provide the ability to provide confidentiality protection between multiple nodes, i.e. multiple transit devices and a NVE.
- o SEC-GEN-6: Geneve security mechanism MUST provide the ability to pad a Geneve packet.
- o SEC-GEN-7: Geneve security mechanism MUST provide the ability to send dummy packets.

#### 5.2. Protecting Against Traffic Injection

Traffic injection from a rogue non legitimate NVO3 Geneve overlay device or a rogue underlay transit device can target an NVE, a transit underlay device or a Tenant System. Targeting a Tenant's System requires a valid MAC and IP addresses of the Tenant's System.

When traffic between tenants is not protected, the rogue device may forward the modified packet over a valid (authenticated) Geneve Header. The crafted packet may for example, include a specifically crafted application payload for a specific Tenant Systems

application, with the intention to load the tenant specific application. Tenant's System may provide integrity protection of the inner payload by protect their communications using for example IPsec/ESP, IPsec/AH [RFC4302], TLS or DTLS. Such protection protects at various layers the Tenants from receiving spoofed packets, as any injected packet is expected to be discarded by the destination Tenant's System. Note IPsec/ESP with NULL encryption may be used to authenticate-only the layers above IP in which case the IP header remains unprotected. However IPsec/AH enables the protection of the entire IP packet, including the IP header. As a result, when Geneve encapsulates IP packets the Tenant has the ability to integrity protect the IP packet on its own, without relying on the Geneve overlay network. On the other hand, L2 layers remains unprotected. As encryption is using authenticated encryption, authentication may also be provided via encryption. At the time of writing the document DTLS 1.3 [I-D.ietf-tls-dtls13] is still a draft document and TLS 1.3 does not yet provide the ability for authenticate only the traffic. As such it is likely that the use of DTLS1.3 may not involve authentication-only cipher suites. Similarly to confidentiality protection, integrity protection may be handled either entirely by the Geneve Overlay Provider, or partially by the Tenant or handled by both the Tenant and the Geneve Overlay Provider.

In addition to confidentiality protection of the inner payload, integrity protection also prevents the Tenant System from receiving illegitimate packets that may disrupt the Tenant's System performance. The Geneve overlay network need to prevent the overlay to be used as a vector to spoof packets being steered to the Tenant's system. As a result, the Overlay Network Provider needs to ensure that inner packets steered to the Tenant's network are only originating from one Tenant System and not from an outsider using the Geneve Overlay to inject packets to one virtual network. As such, the destination NVE MUST be able to authenticate the incoming Geneve packets from the source NVE. This may be performed by the NVE authenticating the full Geneve Packet. When the Geneve Overlay wants to take advantage of the authentication performed by the Tenant System, the NVE should be able to perform some checks between the Geneve Header and the inner payload. Suppose two Geneve packets are composed of a Geneve Header (H1, and H2) and a inner payload (P1 and P2). Suppose H1, H2, P1 and P2 are authenticated. The replacement of P2 by P1 by an attacker will be detected by the NVE only if there is a binding between H2 and P2. Such integrity protection is handled by the Geneve Overlay Provider.

While traffic injection may target the Tenant's virtual network or a specific Tenant System, traffic injection may also target the Geneve Overlay Network by injecting Geneve Options that will affect the processing of the Geneve Packet. Updating the Geneve header and

option parameters such as setting an OAM bit, adding bogus option TLVs, or setting a critical bit, may result in different processing behavior, that could greatly impact performance of the overlay network and the underlay infrastructure and thus affect the tenants traffic delivery. As such, the Geneve Overlay should provide integrity protection of the Geneve Information present in the Geneve Header to guarantee Geneve processing is not altered.

The Geneve architecture considers transit devices that may process some Geneve Options. More specifically, a Geneve packet may have A subset of Geneve Information of the Geneve Header (excluding Geneve Options) as well as a set of zero, one or multiple of Geneve Options accessed by one or more transit devices. This information needs to be authenticated by a transit device while other options may be authenticated by other transit devices or the tunnel endpoint. The integrity protection is handled by the Geneve Overlay Provider and authentication MUST be performed prior any processing.

#### 5.2.1. Operational Security Requirements

A secure deployment of a Geneve overlay must fulfill the requirement below:

- o SEC-OP-4: A secure deployment of a Geneve overlay MUST provide the capability authenticate the inner payload when encryption is not provided. A Geneve overlay provider MAY disable this capability for example when this is performed by the Tenant System and that level of integrity is believed to be sufficient. In order to provide additional protection to traffic already protected by the Tenant the Geneve network operator MAY partially protect the unprotected part of the inner payload.
- o SEC-OP-5: A secure deployment of a Geneve overlay MUST evaluate the risk associated to a change of the Geneve Outer Header, Geneve Header (excluding Geneve Options) and Geneve Option. When a risk analysis concludes that the risk is too high, this piece of information MUST be authenticated.
- o SEC-OP-6: A secure deployment of a Geneve overlay SHOULD authenticate communications between NVE to protect the Geneve Overlay infrastructure as well as the Tenants System's communications (Geneve Packet). A Geneve overlay provider MAY disable authentication of the inner packet and delegates it to the Tenant Systems when communications between Tenant's System is secured. This is NOT RECOMMENDED. Instead, it is RECOMMENDED that mechanisms binds the inner payload to the Geneve Header. To prevent injection between virtualized network, it is strongly

RECOMMENDED that at least the Geneve Header without Geneve Options is authenticated.

- o SEC-OP-7: A secure deployment of a Geneve overlay SHOULD NOT process data prior authentication. If that is not possible, the Geneve overlay provider SHOULD evaluate its impact.

### 5.2.2. Geneve Security Requirements

A Geneve security mechanism must fulfill the requirements below:

- o SEC-GEN-8: Geneve security mechanism MUST provide the capability to authenticate the inner payload.
- o SEC-GEN-9: Geneve security mechanism SHOULD provide the capability to partially authenticate the inner payload header.
- o SEC-GEN-10: Geneve security mechanism MUST provide the capability to authenticate a single or a set of options while leave other Geneve Option unauthenticated. Reversely, a Geneve security mechanism MUST be able to leave a Geneve option unauthenticated, while encrypting the others.
- o SEC-GEN-11: Geneve security mechanism MUST provide means to authenticate the information of Geneve Header (Geneve Option excluded). Reversely, a Geneve security mechanism MUST be able to leave unauthenticated Geneve header information (Geneve Options excluded) while authenticating the other.
- o SEC-GEN-12: Geneve Security mechanism MUST provide means for a tunnel endpoint (NVE) to authenticate data prior it is being processed.
- o SEC-GEN-13: Geneve Security mechanism MUST provide means for a transit device to authenticate data prior it is being processed.

### 5.3. Protecting Against Traffic Redirection

A rogue device of the NVO3 overlay Geneve network or the underlay network may redirect the traffic from a virtual network to the attacker for passive or active attacks. If the rogue device is in charge of securing the Geneve packet, then Geneve security mechanisms are not intended to address this threat. More specifically, a rogue source NVE will still be able to redirect the traffic in clear text before protecting ( and encrypting the packet). A rogue destination NVE will still be able to redirect the traffic in clear text after decrypting the Geneve packets. The same occurs with a rogue transit that is in charge of encrypting and decrypting a Geneve Option,

Geneve Option or any information. The security mechanisms are intended to protect a Geneve information from any on path node. Note that modern cryptography recommend the use of authenticated encryption. This section assumes such algorithms are used, and as such encrypted packets are also authenticated.

To prevent an attacker located in the middle between the NVEs and modifying the tunnel address information in the data packet header to redirect the data traffic, the solution needs to provide confidentiality protection for data traffic exchanged between NVEs.

Requirements are similar as those provided in section Section 5.1 to mitigate sniffing attacks and those provided in section Section 5.2 to mitigate traffic injection attacks.

#### 5.4. Protecting Against Traffic Replay

A rogue device of the NVO3 overlay Geneve network or the underlay network may replay a Geneve packet, to load the network and/or a specific Tenant System with a modified Geneve payload. In some cases, such attacks may target an increase of the tenants costs.

When traffic between Tenant System is not protected against anti-replay. A packet even authenticated can be replayed. DTLS and IPsec provides anti replay mechanisms, so it is unlikely that authenticated Tenant's traffic is subject to replay attacks.

Similarly to integrity protection, the Geneve Overlay Provider should prevent the overlay to be used to replay packet to the Tenant's System. In addition, similarly to integrity protection, the Geneve Overlay network may also be a target of a replay attack, and NVE as well as transit devices should benefit from the same protection.

Given the proximity between authentication and anti-replay mechanisms and that most authentication mechanisms integrates anti-replay attacks, we RECOMMEND that authentication contains an anti-replay mechanisms.

##### 5.4.1. Geneve Security Requirements

A secure deployment of a Geneve overlay must fulfill the requirement below:

- o SEC-OP-8: A secure deployment of a Geneve overlay MUST evaluate the communications subject to replay attacks. Communications that are subject to this attacks MUST be authenticated with an anti replay mechanism. Note that when partial authentication is provided, the part not covered by the authentication remains a

surface of attack. It is strongly RECOMMENDED that the Geneve Header is authenticated with anti replay protection.

#### 5.4.2. Geneve Security Requirements

A Geneve security mechanism must fulfill the requirements below:

- o SEC-GEN-14: Geneve Security mechanism MUST provide authentication with anti-replay protection.

#### 5.5. Security Management

##### 5.5.1. Operational Security Requirements

A secure deployment of a Geneve overlay must fulfill the requirement below:

- o SEC-OP-9: A secure deployment of a Geneve overlay MUST define the security policies that associates the encryption, and authentication associated to each flow between NVEs.
- o SEC-OP-10: A secure deployment of a Geneve overlay SHOULD define distinct material for each flow. The cryptographic depends on the nature of the flow (multicast, unicast) as well as on the security mechanism enabled to protect the flow.

##### 5.5.2. Geneve Security Requirements

A Geneve security mechanism must fulfill the requirements below:

- o SEC-GEN-15: A Geneve security mechanism MUST be managed via security policies associated for each traffic flow to be protected. Geneve overlay provider MUST be able to configure NVEs with different security policies for different flows. A flow MUST be identified at minimum by the Geneve virtual network identifier and the inner IP and transport headers, and optionally additional fields which define a flow (e.g., inner IP DSCP, IPv6 flow id, Geneve options).
- o SEC-GEN-16: A Geneve security mechanism MUST be able to assign different cryptographic keys to protect the unicast tunnels between NVEs respectively.
- o SEC-GEN-17: A Geneve security mechanisms, when multicast is used, packets, MUST be able to assign distinct cryptographic group keys to protect the multicast packets exchanged among the NVEs within different multicast groups. Upon receiving a data packet, an egress Geneve NVE MUST be able to verify whether the packet is

sent from a proper ingress NVE which is authorized to forward that packet.

## 6. IANA Considerations

There are no IANA consideration for this document.

## 7. Security Considerations

The whole document is about security.

Limiting the coverage of the authentication / encryption provides some means for an attack to craft special packets.

The current document details security requirements that are related to the Geneve protocol. Instead, [I-D.ietf-nvo3-security-requirements] provides generic architecture security requirement upon the deployment of an NVO3 overlay network. It is strongly recommended to read that document as architecture requirements also apply here. In addition, architecture security requirements go beyond the scope of Geneve communications, and as such are more likely to address the security needs upon deploying an Geneve overlay network.

## 8. Appendix

### 8.1. DTLS

This section compares how NVE communications using DTLS meet the security requirements for a secure Geneve overlay deployment. In this example DTLS is used over the Geneve Outer Header and secures the Geneve Header including the Geneve Options and the inner payload.

The use of DTLS MAY fill the security requirements for a secure Geneve deployment. However DTLS cannot be considered as the Geneve security mechanism enabling all Geneve deployments. To ease the reading of the Requirements met by DTLS or IPsec, the requirements list indicates with Y (Yes) when the requirement and N (No) when the requirement is not met. In addition, an explanation is provided on the reasoning. This section is not normative and its purpose is limited to illustrative purpose.

#### 8.1.1. Operational Security Requirements

This section shows how DTLS may secure some Geneve deployments. Some Geneve deployments may not be secured by DTLS, but that does not exclude DTLS from being used.



- o SEC-OP-1 (Y): A deployment using DTLS between NVEs with an non NULL encryption cipher suite will provide confidentiality to the full Geneve Packet which contains the inner payload. As such the use of DTLS meets SEC-OP-1. Note that DTLS does not provide partial encryption and as such the Geneve Overlay Provider may not benefit from the encryption performed by the Tenant if performed, which may result in some portion of the payload being encrypted twice.
- o SEC-OP-2 (Y): A deployment using DTLS between NVEs with an non NULL encryption cipher suite encrypt the Geneve Packet which includes the Geneve Header and associated metadata. Only the UDP port is leaked which could be acceptable. As such, the use of DTLS meets SEC-OP-2.
- o SEC-OP-3 (Y/N): A deployment using DTLS between NVEs will not be able to send dummy packets or pad Geneve Packet unless this is managed by the Geneve packet itself. DTLS does not provide the ability to send dummy traffic, nor to pad. As a result DTLS itself does not meet this requirement. This requirement may be met if handled by the Geneve protocol. As such SEC-OP-3 may not be met for some the deployment. However, it is not a mandatory requirement and as such it is likely that the use of DTLS SEC-OP-3 is met.
- o SEC-OP-4 (Y): Similarly to SEC-OP-1, A deployment using DTLS between NVEs provides integrity protection to the full Geneve Packet which includes the inner payload. As such the use of DTLS meets SEC-OP-4. Note that DTLS 1.2 provides integrity-only cipher suites while DTLS 1.3 does not yet. As a result, the use of DTLS 1.3 may provide integrity protection using authenticated encryption.
- o SEC-OP-5 (Y): Similarly to SEC-OP-2, A deployment using DTLS between NVE authenticates the full Geneve Packet which includes the Geneve Header. Only the UDP port is left unauthenticated. As such, the use of DTLS meets SEC-OP-5.
- o SEC-OP-6 (Y): A deployment using DTLS between NVE authenticates NVE-to-NVE communications and the use of DTLS meets SEC-OP-6.
- o SEC-OP-7 (Y/N): A deployment using DTLS between NVEs is not compatible with a Geneve architecture that includes transit devices. When the DTLS session uses a non NULL encryption cipher suite, the transit device will not be able to access it. When the NULL encryption cipher suite is used, the transit device may be able to access the data, but will not be able to authenticate it prior to processing the packet. As such the use of DTLS only

meets SEC-OP-7 for deployment that do not include any transit devices.

- o SEC-OP-8 (Y): A deployment using DTLS between NVEs provides anti-replay protection and so, the use of DTLS meets SEC-OP-8.
- o SEC-OP-9 (Y/N): DTLS does not define any policies. Instead DTLS process is bound to an UDP socket. As such handling of flow policies is handled outside the scope of DTLS. As such SEC-OP-9 is met outside the scope of DTLS.
- o SEC-OP-10 (N): DTLS session may be established with specific material, as such it is possible to assign different material for each flow. However, the binding between flow and session is performed outside the scope of DTLS. In addition, DTLS does not support multicast. As such, the use of DTLS may only meets SEC-OP-10 in the case of unicast communications.

#### 8.1.2. Geneve Security Requirements

This section shows that DTLS cannot be used as a generic Geneve security mechanism to secure Geneve deployments. A Geneve security mechanism would need to meet all SEC-GEN requirements.

- o SEC-GEN-1 (Y): A deployment using DTLS between NVEs with an non NULL encryption cipher suite will provide confidentiality to the full Geneve Packet which contains the inner payload. As such the use of DTLS meets SEC-GEN-1.
- o SEC-GEN-2 (Y): A deployment using DTLS between NVEs with an non NULL encryption cipher suite will not be able to partially encrypt the inner payload header. However such requirement is not set a mandatory so the use of DTLS meets SEC-GEN-2
- o SEC-GEN-3 (N): A deployment using DTLS between NVEs with an non NULL encryption cipher suite encrypt the Geneve Packet which includes the Geneve Header and all Geneve Options. However DTLS does not provides any means to selectively encrypt or leave in clear text a subset of Geneve Options. As a result the use of DTLS does not meet SEC-GEN-3.
- o SEC-GEN-4 (N): A deployment using DTLS between NVEs with an non NULL encryption cipher suite encrypt the Geneve Packet which includes the Geneve Header and all Geneve Options. However, DTLS does not provides means to selectively encrypt some information of the Geneve Header. As such the use of DTLS does not meet SEC-GEN-5.

- o SEC-GEN-5 (N): A deployment using DTLS between NVEs with an non NULL encryption cipher suite provides end-to-end security between the NVEs and as such does not permit the interaction of one or multiple on-path transit devices. As such the use of DTLS does not meet SEC-GEN-5.
- o SEC-GEN-6 (N): A deployment using DTLS between NVEs with an non NULL encryption cipher suite does not provide padding facilities. This requirements is not met by DTLS itself and needs to be handled by Geneve and specific options. As a result, the use of DTLS does not meet SEC-GEN-6
- o SEC-GEN-7 (N): A deployment using DTLS between NVEs with an non NULL encryption cipher suite does not provide the ability to send dummy packets. This requirements is not met by DTLS itself and needs to be handled by Geneve and specific options. As a result, the use of DTLS does not meet SEC-GEN-7.
- o SEC-GEN-8 (Y): A deployment using DTLS between NVEs with an non NULL encryption cipher suite or a NULL encryption cipher suite provide authentication of the inner payload. As such the use of DTLS meets SEC-GEN-8.
- o SEC-GEN-9 (Y): A deployment using DTLS between NVEs does not provide the ability to partially authenticate the inner payload header. However such requirement is not set a mandatory so the use of DTLS meets SEC-GEN-9
- o SEC-GEN-10 (N): A deployment using DTLS between NVEs authenticates the Geneve Packet which includes the Geneve Header and all Geneve Options. However, DTLS does not provides means to selectively encrypt some information of the Geneve Header. As such the use of DTLS meets SEC-GEN-10.
- o SEC-GEN-11 (N): A deployment using DTLS between NVEs authenticates the Geneve Packet which includes the Geneve Header and all Geneve Options. However, DTLS does not provides means to selectively authenticate some information of the Geneve Header. As such the use of DTLS does not meet SEC-GEN-11.
- o SEC-GEN-12 (Y): A deployment using DTLS between NVEs authenticates the data prior the data is processed by the NVE. As such, the use of DTLS meets SEC-GEN-12.
- o SEC-GEN-13 (N): A deployment using DTLS between NVEs authenticates the data when the tunnel reaches the NVE. As a result the transit device is not able to authenticate the data prior accessing it and the use of DTLS does not meet SEC-GEN-13.

- o SEC-GEN-14 (Y): DTLS provides anti-replay mechanism as such, the use of DTLS meets SEC-GEN-14.
- o SEC-GEN-15 (N): DTLS itself does not have a policy base mechanism. As a result, the classification of the flows needs to be handled by a module outside DTLS. In order to meet SEC-GEN-15 further integration is needed and DTLS in itself cannot be considered as meeting SEC-GEN-15.
- o SEC-GEN-16 (Y): DTLS is able to assign various material to each flows, as such the use of DTLS meets SEC-GEN-16.
- o SEC-GEN-17 (N): DTLS does not handle multicast communications. As such the use of DTLS does not meet SEC-GEN-17.

## 8.2. IPsec

This section compares how NVE communications using IPsec/ESP or IPsec/AH meet the security requirements for a secure Geneve overlay deployment. In this example secures the Geneve IP packet including Outer IP header, the Geneve Outer Header, the Geneve Header including Geneve Options and the inner payload.

The use of IPsec/ESP or IPsec/AH share most of the analysis performed for DTLS. The main advantages of using IPsec would be that IPsec supports multicast communications and natively supports flow based security policies. However, the use of these security policies in a context of Geneve is not natively supported.

As a result, the use of IPsec MAY fill the security requirements for a secure Geneve deployment. However IPsec cannot be considered as the Geneve security mechanism enabling all Geneve deployments.

### 8.2.1. Operational Security Requirements

This section shows how IPsec may secure some Geneve deployments. Some Geneve deployments may not be secured by IPsec, but that does not exclude IPsec from being used.

- o SEC-OP-1 (Y): A deployment using IPsec/ESP between NVEs with an non NULL encryption will provide confidentiality to the full Outer IP payload of the Geneve Packet which contains the inner payload. As a result, such deployments meet SEC-OP-1. Note that IPsec/ESP does not provide partial encryption and as such the Geneve Overlay Provider may not benefit from the encryption performed by the Tenant if performed, which may result in some portion of the payload being encrypted twice.

- o SEC-OP-2 (Y): A deployment using IPsec/ESP between NVEs with a non NULL encryption encrypts the Outer IP payload Geneve IP Packet which includes the Geneve Header and associated information. As such SEC-OP-2 is met.
- o SEC-OP-3 (Y): A deployment using IPsec/ESP between NVEs will be able to send dummy packets or pad Geneve Packet. As such OP-SEC-3 is met.
- o SEC-OP-4 (Y): Similarly to SEC-OP-1, A deployment using IPsec/ESP or IPsec/AH between NVEs provides integrity protection to the full Geneve Packet which includes the inner payload. As such SEC-OP-4 is met.
- o SEC-OP-5 (Y): Similarly to SEC-OP-2, A deployment using IPsec/ESP or IPsec/AH between NVE authenticates the full Geneve Packet which includes the Geneve Header. As such SEC-OP-5 is met as well.
- o SEC-OP-6 (Y): A deployment using IPsec/ESP or IPsec/AH between NVE authenticates NVE-to-NVE communications and SEC-OP-6 is met.
- o SEC-OP-7 (Y/N): A deployment using IPsec between NVEs is not compatible with a Geneve architecture that includes transit devices. When IPsec/ESP with a non NULL encryption is used, the transit device will not be able to access it. When IPsec/AH or IPsec/ESP with the NULL encryption is used, the transit device may be able to access the data, but will not be able to authenticate it prior to processing the packet. As SEC-OP-7 is only met for deployment that do not include any transit devices.
- o SEC-OP-8 (Y): A deployment using IPsec between NVEs provides anti-replay protection and so meets SEC-OP-8.
- o SEC-OP-9 (Y/N): IPsec enables the definition of security policies. As such IPsec is likely to handle a per flow security. However the traffic selector required for Geneve flows may not be provided natively by IPsec. As such Sec-OP-9 is only partially met.
- o SEC-OP-10 (Y): IPsec session may be established with specific material, as such it is possible to assign different material for each flow. In addition IPsec supports multicats communications. As such SEC-OP-10 is met.

#### 8.2.2. Geneve Security Requirements

This section shows that IPsec cannot be used as a generic Geneve security mechanism to secure Geneve deployments. A Geneve security mechanism would need to meet all SEC-GEN requirements.

- o SEC-GEN-1 (Y): A deployment using IPsec/ESP between NVEs with an non NULL encryption provide confidentiality to the full Geneve Packet which contains the inner payload. As such IPsec/ESP meets SEC-GEN-1.
- o SEC-GEN-2 (Y): A deployment using IPsec/ESP between NVEs with an non NULL encryption will not be able to partially encrypt the inner payload header. However such requirement is not set a mandatory so IPsec/ESP meets SEC-GEN-2
- o SEC-GEN-3 (N): A deployment using IPsec between NVEs with an non NULL encryption encrypts the Outer IP payload of the Geneve Packet which includes the Geneve Header and all Geneve Options. However IPsec/ESP does not provides any means to selectively encrypt or leave in clear text a subset of Geneve Options. As a result SEC-GEN-3 is not met.
- o SEC-GEN-4 (N): A deployment using IPsec/ESP between NVEs with an non NULL encryption encrypts the Geneve Packet which includes the Geneve Header and all Geneve Options. However, IPsec/ESP does not provides means to selectively encrypt some information of the Geneve Header. As such SEC-GEN-5 is not met.
- o SEC-GEN-5 (N): A deployment using IPsec between NVEs with an non NULL encryption provides end-to-end security between the NVEs and as such does not permit the interaction of one or multiple on-path transit devices. As such IPsec/ESP does not meet SEC-GEN-5.
- o SEC-GEN-6 (Y): A deployment using IPsec/ESP between NVEs with an non NULL encryption provides padding facilities and as such IPsec/ESP meets SEC-GEN-6.
- o SEC-GEN-7 (Y): A deployment using IPsec between NVEs with an non NULL encryption cipher provides the ability to send dummy packets. As such IPsec/ESP meets SEC-GEN-7.
- o SEC-GEN-8 (Y): A deployment using IPsec/ESP or IPsec/AH authenticates the inner payload. As such SEC-GEN-8 is met.
- o SEC-GEN-9 (Y): A deployment using IPsec/AH or IPsec/ESP between NVEs does not provide the ability to partially authenticate the inner payload header. However such requirement is not set a mandatory so IPsec meets SEC-GEN-9
- o SEC-GEN-10 (N): A deployment using IPsec/ESP or IPsec/AH between NVEs authenticates the Geneve Packet which includes the Geneve Header and all Geneve Options. However, IPsec does not provides

means to selectively encrypt some information of the Geneve Header. As such SEC-GEN-10 is not met.

- o SEC-GEN-11 (N): A deployment using IPsec/ESP or IPsec/AH between NVEs authenticates the Geneve Packet which includes the Geneve Header and all Geneve Options. However, IPsec does not provides means to selectively authenticate some information of the Geneve Header. As such SEC-GEN-11 is not met.
- o SEC-GEN-12 (Y): A deployment using IPsec/ESP or IPsec/AH between NVEs authenticates the data prior the data is processed by the NVE. As such SEC-GEN-12 is met.
- o SEC-GEN-13 (N): A deployment using IPsec/ESP or IPsec/AH between NVEs authenticates the data when the tunnel reaches the NVE. As a result the transit device is not able to authenticate the data prior accessing it and SEC-GEN-13 is not met.
- o SEC-GEN-14 (Y): IPsec/ESP and IPsec/AH provides anti-replay mechanism as such SEC-GEN-14 is met.
- o SEC-GEN-15 (N): IPsec is a policy base architecture. As a result, the classification of the flows needs to be handled by IPsec. However, the traffic selector available are probably not those required by Geneve and further integration is needed. As such SEC-GEN-15 is not met.
- o SEC-GEN-16 (Y): IPsec is able to assign various material to each flows, as such SEC-GEN-16 is met.
- o SEC-GEN-17 (Y): IPsec handles mutlicast communications. As such SEC-GEN-17 is met.

## 9. Acknowledgments

We would like to thank Ilango S Ganaga, Magnus Nystroem for their useful reviews and clarifications as well as Matthew Bocci, Sam Aldrin and Ignas Bagdona for moving the work forward.

## 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/info/rfc2119>>.

- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<https://www.rfc-editor.org/info/rfc4301>>.
- [RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <<https://www.rfc-editor.org/info/rfc4302>>.
- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <<https://www.rfc-editor.org/info/rfc4303>>.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <<https://www.rfc-editor.org/info/rfc6347>>.
- [RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <<https://www.rfc-editor.org/info/rfc7258>>.
- [RFC7365] Lasserre, M., Balus, F., Morin, T., Bitar, N., and Y. Rekhter, "Framework for Data Center (DC) Network Virtualization", RFC 7365, DOI 10.17487/RFC7365, October 2014, <<https://www.rfc-editor.org/info/rfc7365>>.
- [RFC7516] Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", RFC 7516, DOI 10.17487/RFC7516, May 2015, <<https://www.rfc-editor.org/info/rfc7516>>.
- [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <<https://www.rfc-editor.org/info/rfc7525>>.
- [RFC8014] Black, D., Hudson, J., Kreeger, L., Lasserre, M., and T. Narten, "An Architecture for Data-Center Network Virtualization over Layer 3 (NVO3)", RFC 8014, DOI 10.17487/RFC8014, December 2016, <<https://www.rfc-editor.org/info/rfc8014>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.



- [RFC8221] Wouters, P., Migault, D., Mattsson, J., Nir, Y., and T. Kivinen, "Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH)", RFC 8221, DOI 10.17487/RFC8221, October 2017, <<https://www.rfc-editor.org/info/rfc8221>>.
- [RFC8247] Nir, Y., Kivinen, T., Wouters, P., and D. Migault, "Algorithm Implementation Requirements and Usage Guidance for the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 8247, DOI 10.17487/RFC8247, September 2017, <<https://www.rfc-editor.org/info/rfc8247>>.
- [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/info/rfc8446>>.

## 10.2. Informative References

- [I-D.ietf-nvo3-geneve]  
Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-09 (work in progress), February 2019.
- [I-D.ietf-nvo3-security-requirements]  
Hartman, S., Zhang, D., Wasserman, M., Qiang, Z., and M. Zhang, "Security Requirements of NVO3", draft-ietf-nvo3-security-requirements-07 (work in progress), June 2016.
- [I-D.ietf-tls-dtls13]  
Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", draft-ietf-tls-dtls13-30 (work in progress), November 2018.

## Authors' Addresses

Daniel Migault  
Ericsson  
8275 Trans Canada Route  
Saint Laurent, QC 4S 0B6  
Canada  
  
EMail: [daniel.migault@ericsson.com](mailto:daniel.migault@ericsson.com)

Sami Boutros  
VMware, Inc.

EMail: [boutros@vmware.com](mailto:boutros@vmware.com)<

Dan Wings  
VMware, Inc.

EMail: [dwing@vmware.com](mailto:dwing@vmware.com)

Suresh Krishnan  
Kaloom

EMail: [suresh@kaloom.com](mailto:suresh@kaloom.com)

NVO3 Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: January 6, 2020

G. Mirsky  
X. Min  
ZTE Corp.  
S. Boutros  
VmWare, Inc.  
D. Black  
Dell EMC  
July 5, 2019

OAM for use in GENEVE  
draft-mmabb-nvo3-geneve-oam-00

Abstract

This document defines encapsulation for active Operations, Administration, and Maintenance protocols in Geneve protocol. Also, the format and operation of the Echo Request and Echo Reply mechanism in Geneve are defined.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 6, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|   |   |
|---|---|
| 1. Introduction . . . . .                                   | 2 |
| 1.1. Conventions used in this document . . . . .            | 3 |
| 1.1.1. Terminology . . . . .                                | 3 |
| 1.1.2. Requirements Language . . . . .                      | 3 |
| 2. OAM Protocols Encapsulation in Geneve Networks . . . . . | 3 |
| 2.1. OAM Encapsulation in Geneve . . . . .                  | 4 |
| 3. Associated Channel in Geneve Networks . . . . .          | 5 |
| 4. Associate Channel Header in Geneve . . . . .             | 5 |
| 4.1. Use of the Geneve ACh Header in Active OAM . . . . .   | 6 |
| 5. Echo Request and Echo Reply in Geneve Tunnel . . . . .   | 6 |
| 6. IANA Considerations . . . . .                            | 6 |
| 6.1. Geneve Associated Channel Protocol Types . . . . .     | 6 |
| 7. Security Considerations . . . . .                        | 7 |
| 8. Acknowledgment . . . . .                                 | 7 |
| 9. References . . . . .                                     | 7 |
| 9.1. Normative References . . . . .                         | 7 |
| 9.2. Informative References . . . . .                       | 8 |
| Authors' Addresses . . . . .                                | 8 |

## 1. Introduction

Geneve [I-D.ietf-nvo3-geneve] is intended to support various scenarios of network virtualization. In addition to carrying multi-protocol payload, e.g., Ethernet, IPv4/IPv6, the Geneve message includes metadata. Operations, Administration, and Maintenance (OAM) protocols support fault management and performance monitoring functions necessary for comprehensive network operation. Active OAM protocols, as defined in [RFC7799], use specially constructed packets, that are injected into the network. To ensure that the measured performance metric or the detected failure of the transport layer are related to the particular Geneve flow, it is critical that these test packets share fate with overlay data packets when traversing the underlay network.

This document describes several options for encapsulation of active OAM protocols in Geneve. These are intended to facilitate the discussion among experts and all interested in both OAM and Geneve subjects. The goal of such analysis is the selection of one encapsulation method and providing

Also, a set of generic requirements for active OAM protocols in Geneve overlay network introduced in this document. These should be

used in selecting the most suitable encapsulation for active OAM in Geneve.

## 1.1. Conventions used in this document

### 1.1.1. Terminology

CC Continuity Check

CV Connectivity Verification

FM Fault Management

GAL Generic Associated Channel Label

G-ACh Generic Associated Channel Header

Geneve Generic Network Virtualization Encapsulation

NVO3 Network Virtualization Overlays

OAM Operations, Administration, and Maintenance

ACh Associated Channel

### 1.1.2. Requirements Language

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.

## 2. OAM Protocols Encapsulation in Geneve Networks

OAM protocols, whether it is part of fault management or performance monitoring, intended to provide reliable information that can be used to detect a failure, identify the defect, localize it, thus helping to apply corrective actions to minimize the negative impact on service. Several OAM protocols will be used to perform these functions, protocols that require demultiplexing at the receiving instance of Geneve. To improve the accuracy of the correlation between the condition experienced by the monitored Geneve tunnel and the state of the OAM protocol the OAM encapsulation is required to comply with the following requirements:

REQ#1: Geneve OAM packets SHOULD share the fate with data traffic of the monitored Geneve tunnel, i.e., be in-band with the

monitored traffic, follow precisely the same overlay and transport path as packets with data payload, in the forward direction, i.e. from ingress toward egress endpoint(s) of the OAM test.

REQ#2: Encapsulation of OAM control message and data packets in underlay network MUST be indistinguishable from the underlay network forwarding point of view.

REQ#3: Presence of OAM control message in Geneve packet MUST be unambiguously identifiable.

REQ#4: It MUST be possible to express entropy for underlay Equal Cost Multipath in Geneve encapsulation to avoid using data packet content by underlay transient nodes.

## 2.1. OAM Encapsulation in Geneve

One of the options is to use IP/UDP encapsulation for active OAM. In this case OAM protocols are identified by destination UDP port number. This approach is well-known and has been used, for example, in MPLS networks. To use IP/UDP encapsulation for an active OAM protocol the Protocol Type field of the Geneve header MUST be set to IPv4 (0x0800) or IPv6 (0x86DD) value. But extra IP/UDP headers that immediately follow the Geneve header adds to processing of OAM message, further disassociates OAM message from the Geneve header, all of which may cause false negative or positive failure reports. Also, the additional IP/UDP header adds noticeable overhead, especially if the underlay is the IPv6 network.

Another option is to use the Protocol Type field to demultiplex an active OAM protocols directly. Such method avoids the use of additional intermediate header but requires that each active OAM protocol be assigned unique identifier from the Ether Types registry maintained by IANA.

The alternative to using the Protocol Type directly is to use a shim that, in turn, identifies the OAM Protocol and, optionally, includes additional information. [RFC5586] defines how the Generic Associated Channel Label (GAL) can be used to identify that the Associated Channel Header (ACH), defined in [RFC4385], immediately follows the Bottom-of-the-Stack label. Thus the MPLS Generic Associated Channel can be identified, and protocols are demultiplexed based on the value of the Channel Type field. Number of channel types, e.g., for continuity check and performance monitoring, already have been defined and are listed in IANA MPLS Generalized Associated Channel (G-ACH) Types (including Pseudowire Associated Channel Types) registry. To use this approach, the value of the Protocol Type field in the Geneve header MUST be set to MPLS. The Geneve header MUST be

immediately followed by the GAL label with the S flag set to indicate that GAL is the Bottom-of-the-stack label. Then ACH MUST follow the GAL label and the value of the Channel Type identifies which of active OAM protocols being encapsulated in the packet.

Lastly, an associated channel can be defined directly for a Geneve tunnel. This document defines the shim for Geneve is presented in Figure 1 to demultiplex Geneve OAM protocols without much of the overhead. The value of the Protocol Type MAY be set to 0x8902, the value assigned to IEEE 802.lag Connectivity Fault Management protocol as part of [IEEE.8021Q] and shared by ITU-T G.8013/Y.1731 OAM functions and mechanisms for Ethernet-based networks [ITU-T.1731]. Alternatively, the new value MAY be requested from the Ether Types registry.

### 3. Associated Channel in Geneve Networks

An associated channel in the Geneve network is the channel that, by using the same encapsulation as user traffic, follows the same path through the underlay network as user traffic. In other words, the associated channel is in-band with user traffic. Creating the notion of the associated channel (ACh) in the Geneve network ensures that packets of active OAM protocols carried in the ACh are in-band with user traffic.

#### 4. Associate Channel Header in Geneve

ACh Header immediately follows the Geneve header defined in [I-D.ietf-nvo3-geneve] and identifies the type of message carried over the Geneve ACh. The format of the Geneve ACh Header is:

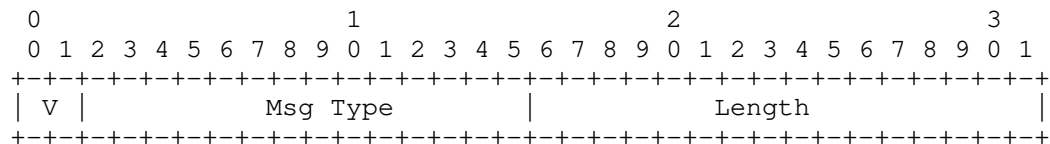


Figure 1: Format of the Associated Channel Header in Geneve Network

The ACh Header consists of the following fields:

- o V - two bits long field indicates the current version of the ACh Header. The current value is 0b00;
- o Msg Type - 14 bits long field identifies a protocol. In the case of active OAM, these could be Echo Request/Reply, BFD, Performance Measurement;

- o Length - two octets long field that is the length of the packet in octets;

#### 4.1. Use of the Geneve ACh Header in Active OAM

Active OAM methods, whether used for fault management or performance monitoring, generate dedicated test packets [RFC7799]. Format of an OAM test packet in Geneve network presented in Figure 2.

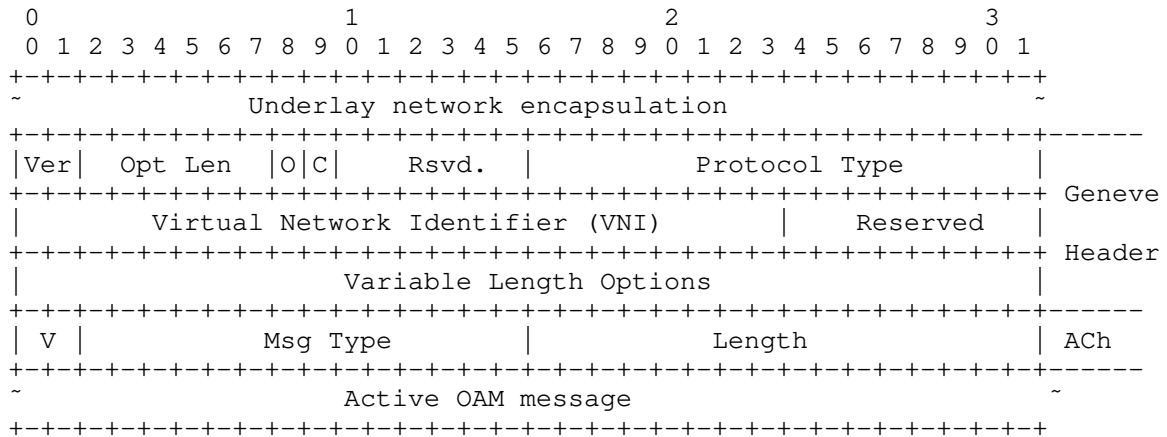


Figure 2: Geneve OAM Header in Active OAM Packet

#### 5. Echo Request and Echo Reply in Geneve Tunnel

[Ed.note] Will be expanded in the future versions.

#### 6. IANA Considerations

IANA is requested to create a new registry called "Geneve Associated Channel".

##### 6.1. Geneve Associated Channel Protocol Types

IANA is requested to create new sub-registry called "Geneve Associated Channel Protocol Types" in the "Geneve Associated Channel" registry. All code points in the range 1 through 15615 in this registry shall be allocated according to the "IETF Review" procedure as specified in [RFC8126]. Remaining code points are allocated according to Table 1:



| Value         | Description  | Reference               |
|---------------|--------------|-------------------------|
| 0             | Reserved     |                         |
| 1 - 15615     | Unassigned   | IETF Review             |
| 15616 - 16127 | Unassigned   | First Come First Served |
| 16128 - 16143 | Experimental | This document           |
| 16144 - 16382 | Private Use  | This document           |
| 16383         | Reserved     | This document           |

Table 1: Geneve OAM Protocol type

## 7. Security Considerations

TBD

## 8. Acknowledgment

TBD

## 9. References

### 9.1. Normative References

[I-D.ietf-nvo3-geneve]

Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-13 (work in progress), March 2019.

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

[RFC4385] Bryant, S., Swallow, G., Martini, L., and D. McPherson, "Pseudowire Emulation Edge-to-Edge (PWE3) Control Word for Use over an MPLS PSN", RFC 4385, DOI 10.17487/RFC4385, February 2006, <<https://www.rfc-editor.org/info/rfc4385>>.

[RFC5586] Bocci, M., Ed., Vigoureux, M., Ed., and S. Bryant, Ed., "MPLS Generic Associated Channel", RFC 5586, DOI 10.17487/RFC5586, June 2009, <<https://www.rfc-editor.org/info/rfc5586>>.

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

## 9.2. Informative References

- [IEEE.8021Q]  
IEEE, "IEEE Standard for Local and metropolitan area networks -- Bridges and Bridged Networks", IEEE Std 802.1Q, December 2014.
- [ITU-T.1731]  
ITU-T, "Operations, administration and maintenance (OAM) functions and mechanisms for Ethernet-based networks", ITU-T G.8013/Y.1731, August 2015.
- [RFC7799] Morton, A., "Active and Passive Metrics and Methods (with Hybrid Types In-Between)", RFC 7799, DOI 10.17487/RFC7799, May 2016, <<https://www.rfc-editor.org/info/rfc7799>>.
- [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/info/rfc8126>>.

## Authors' Addresses

Greg Mirsky  
ZTE Corp.

Email: [gregimirsky@gmail.com](mailto:gregimirsky@gmail.com)

Xiao Min  
ZTE Corp.

Email: [xiao.min2@zte.com.cn](mailto:xiao.min2@zte.com.cn)

Sami Boutros  
VmWare, Inc.

Email: [sboutros@vmware.com](mailto:sboutros@vmware.com)

David Black  
Dell EMC  
176 South Street  
Hopkinton, MA 01748  
United States of America

Email: [david.black@dell.com](mailto:david.black@dell.com)

NVO3 Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: March 29, 2021

G. Mirsky  
ZTE Corp.  
S. Boutros  
Ciena  
D. Black  
Dell EMC  
S. Pallagatti  
VMware  
September 25, 2020

OAM for use in GENEVE  
draft-mmabb-nvo3-geneve-oam-04

## Abstract

This document lists a set of general requirements for active OAM protocols in the Geneve overlay network. Based on the requirements, IP encapsulation for active Operations, Administration, and Maintenance protocols in Geneve protocol is defined. Considerations for using ICMP and UDP-based protocols are discussed.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 29, 2021.

## Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|   |    |
|---|----|
| 1. Introduction . . . . .   | 2  |
| 1.1. Conventions used in this document . . . . .  | 3  |
| 1.1.1. Acronyms . . . . .   | 3  |
| 1.1.2. Requirements Language . . . . .  | 3  |
| 2. Active OAM Protocols in Geneve Networks . . . . .                                      | 3  |
| 2.1. Defect Detection and Troubleshooting in Geneve Network<br>with Active OAM . . . . .  | 4  |
| 2.2. OAM Encapsulation in Geneve . . . . .  | 6  |
| 3. Echo Request and Echo Reply in Geneve Tunnel . . . . .                                 | 7  |
| 4. IANA Considerations . . . . .  | 8  |
| 5. Security Considerations . . . . .  | 8  |
| 6. Acknowledgments . . . . .  | 8  |
| 7. References . . . . .   | 8  |
| 7.1. Normative References . . . . .   | 8  |
| 7.2. Informative References . . . . .   | 9  |
| Appendix A. Additional Considerations for OAM Encapsulation<br>Method in Geneve . . . . . | 10 |
| Authors' Addresses . . . . .  | 10 |

## 1. Introduction

Geneve [I-D.ietf-nvo3-geneve] is intended to support various scenarios of network virtualization. In addition to carrying multi-protocol payload, e.g., Ethernet, IPv4/IPv6, the Geneve message includes metadata. Operations, Administration, and Maintenance (OAM) protocols support fault management and performance monitoring functions necessary for comprehensive network operation. Active OAM protocols, as defined in [RFC7799], use specially constructed packets that are injected into the network. To ensure that the measured performance metric or the detected failure of the transport layer are related to a particular Geneve flow, it is critical that these test packets share fate with overlay data packets for that flow when traversing the underlay network.

A set of general requirements for active OAM protocols in the Geneve overlay network is listed in Section 2. IP encapsulation conforms to these requirements and is defined as a suitable encapsulation of active OAM protocols in a Geneve overlay network. Note that the IP encapsulation of OAM is applicable to those VNIs that support the use of the necessary values of the Protocol Type field in the Geneve

header, i.e., Ethertypes of IPv4 or IPv6. It does not apply to VNIs that lack that support, e.g., VNIs that only support Ethernet Ethertypes. Analysis and definition of other types of OAM encapsulation in Geneve are outside the scope of this document.

## 1.1. Conventions used in this document

### 1.1.1. Acronyms

CC Continuity Check

CV Connectivity Verification

FM Fault Management

Geneve Generic Network Virtualization Encapsulation

NVO3 Network Virtualization Overlays

OAM Operations, Administration, and Maintenance

VNI Virtual Network Identifier

### 1.1.2. Requirements Language

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.

## 2. Active OAM Protocols in Geneve Networks

OAM protocols, whether part of fault management or performance monitoring, are intended to provide reliable information that can be used to detect a failure, identify the defect and localize it, thus helping to identify and apply corrective actions to minimize the negative impact on service. Several OAM protocols are used to perform these functions; these protocols require demultiplexing at the receiving instance of Geneve. To improve the accuracy of the correlation between the condition experienced by the monitored Geneve tunnel and the state of the OAM protocol the OAM encapsulation is required to comply with the following requirements:

REQ#1: Geneve OAM test packets MUST share the fate with data traffic of the monitored Geneve tunnel, i.e., be in-band with the monitored traffic, follow the same overlay and transport path as

packets with data payload, in the forward direction, i.e. from ingress toward egress endpoint(s) of the OAM test.

An OAM protocol MAY be used to monitor the particular Geneve tunnel as a whole. In that case, test packets could be fate-sharing with a sub-set of tenant flows transported over the Geneve tunnel. If the goal is to monitor the condition experienced by the flow of a particular tenant, the test packets MUST be fate-sharing with that specific flow in the Geneve tunnel. In the latter case, the test packet MUST use the same Geneve encapsulation as the data packet (except for the value in the Protocol Type field [I-D.ietf-nvo3-geneve]), including the value in the Virtual Network Identifier (VNI) field. Both scenarios are discussed in detail in Section 2.1.

REQ#2: Encapsulation of OAM control message and data packets in underlay network MUST be indistinguishable from the underlay network IP forwarding point of view.

REQ#3: Presence of OAM control message in Geneve packet MUST be unambiguously identifiable to Geneve functionality, e.g., at endpoints of Geneve tunnels.

REQ#4: OAM test packets MUST NOT be forwarded to a tenant system.

A test packet generated by an active OAM protocol, either for a defect detection or performance measurement, according to REQ#1, MUST be fate-sharing with the tunnel or data flow being monitored. In an environment where multiple paths through the domain are available, underlay transport nodes can be programmed to use characteristic information to balance the load across known paths. It is essential that test packets follow the same route, i.e., traverses the same set of nodes and links, as a data packet of the monitored flow. Thus, the following requirement to support OAM packet fate-sharing with the data flow:

REQ#5: It MUST be possible to express entropy for underlay Equal Cost Multipath in the Geneve encapsulation of OAM packets.

## 2.1. Defect Detection and Troubleshooting in Geneve Network with Active OAM

Figure 1 presents an example of a Geneve domain. In this section, we consider two scenarios of active OAM being used to detect and localize defects in the Geneve network.

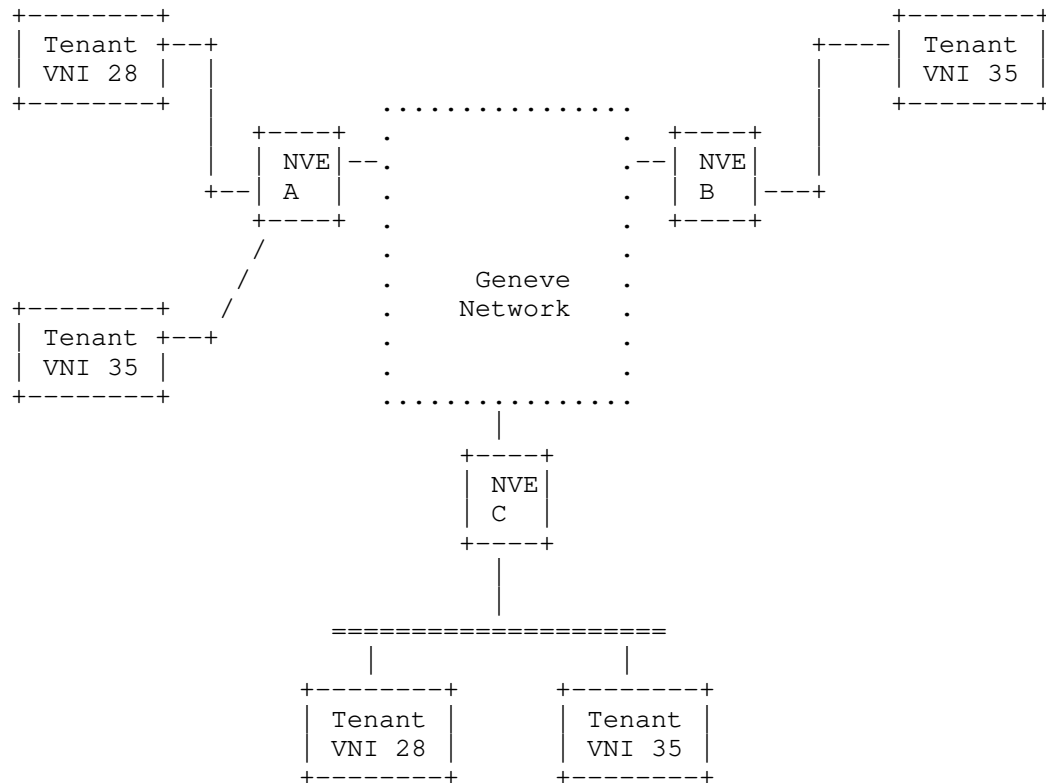


Figure 1: An example of a Geneve domain

In the first case, a communication problem between Network Virtualization Edge (NVE) device A and NVE C was observed. The underlay, e.g., IP network, forwarding is working well but the Geneve connection is unstable for all tenants of NVE A and NVE C. Troubleshooting and localization of the problem can be done irrespective of the VNI value.

In the second case, traffic on VNI 35 between NVE A and NVE B has no problems, as on VNI 28 between NVE A and NVE C. But traffic on VNI 35 between NVE A and NVE C experiences problems, for example, excessive packet loss.

The first case can be detected and investigated using any VNI value, whether it connects tenant systems or not. To conform to REQ#4 (Section 2) OAM test packets could be transmitted on VNI that doesn't have any tenant. That VNI in a Geneve tunnel is dedicated to carrying only control and management data between the tunnel

endpoints, hence communication that uses that VNI is also and referred to as a Geneve control channel. Thus, the control channel of a Geneve tunnel MUST NOT carry tenant data. As no tenants are connected using the control channel, a system that supports this specification, MUST NOT forward a packet received over the control channel to any tenant. A packet received over the control channel MAY be forwarded if and only if it is sent onto the control channel of the concatenated Geneve tunnel. A specific VNI MAY be used to identify the control channel. The value that is associated with this function is referred to as Management VNI. It is RECOMMENDED that the value 1 be used as the default value of Management VNI. Encapsulation of test packets, in this case, is discussed in Section 2.2. The Management VNI SHOULD be terminated on the tenant-facing side of the Geneve encap/decap functionality, not the DC-network-facing side (per definitions in Section 4 [RFC8014]) so that Geneve encap/decap functionality is included in its scope. This approach causes an active OAM packet, e.g., an ICMP echo request, to be decapsulated in the same fashion as any other received Geneve packet. In this example, the resulting ICMP packet is handed to NVE's local management functionality for the processing which generates an ICMP echo reply. The ICMP echo reply is encapsulated in Geneve for forwarding back to the NVE that sent the echo request. One advantage of this approach is that a repeated ping test could detect an intermittent problem in Geneve encap/decap hardware, which would not be tested if the Management VNI were handled as a "special case" at the DC-network-facing interface.

The second case requires that a test packet be transmitted using the VNI value for the traffic that is encountering problems. The encapsulation of the test packet, i.e., inner encapsulation, MUST be the same as the tenant packet's encapsulation in the Geneve tunnel and use the same VNI number. Encapsulation of test packets in this case, is discussed in Section 2.2.

## 2.2. OAM Encapsulation in Geneve

Active OAM in Geneve network uses an IP encapsulation. Protocols such as BFD [RFC5880] or STAMP [RFC8762] use UDP transport. Destination UDP port number in the inner UDP header (Figure 2) identifies the OAM protocol. This approach is well-known and has been used, for example, in MPLS networks [RFC8029]. The UDP source port can be used to provide entropy, e.g., for Equal Cost Multipath. To use IP encapsulation for an active OAM protocol the Protocol Type field of the Geneve header MUST be set to the IPv4 (0x0800) or IPv6 (0x86DD) value.



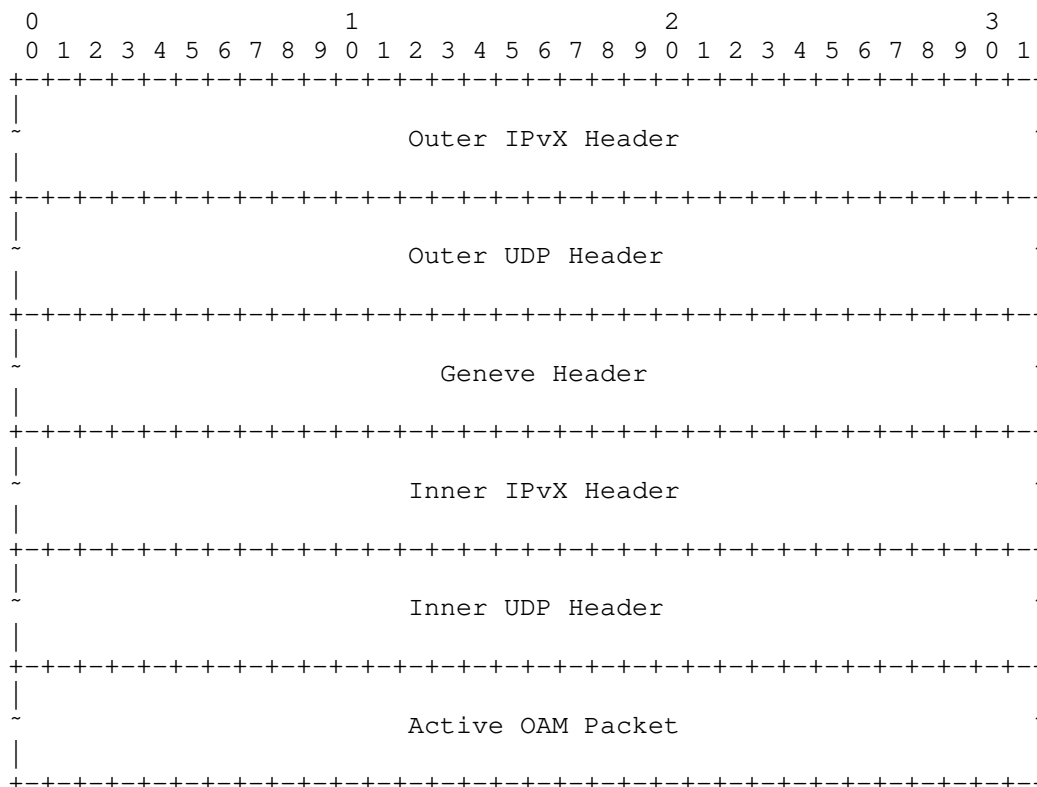


Figure 2: Geneve IP/UDP Encapsulation of an Active OAM Packet

#### Inner IP header:

**Destination IP:** IP address MUST NOT be of one of tenant's IP addresses. The IP address SHOULD be set to the loopback address 127.0.0.1/32 for IPv4, or the loopback address ::1/128 for IPv6 [RFC4291]. Alternatively, the destination IP address MAY be set to VAP's IP address.

**Source IP:** IP address of the originating VAP.

**TTL or Hop Limit:** MUST be set to 255 per [RFC5082].

### 3. Echo Request and Echo Reply in Geneve Tunnel

ICMP and ICMPv6 ([RFC0792] and [RFC4443] respectively) provide required on-demand defect detection and failure localization. ICMP control messages immediately follow the inner IP header encapsulated

in Geneve. ICMP extensions for Geneve networks use mechanisms defined in [RFC4884].

#### 4. IANA Considerations

This document has no requirements for IANA. This section can be removed before the publication.

#### 5. Security Considerations

As part of a Geneve network, active OAM inherits the security considerations discussed in [I-D.ietf-nvo3-geneve]. Additionally, a system MUST provide control to limit the rate of Geneve OAM packets punted to the Geneve control plane for processing in order to avoid overloading that control plane.

#### 6. Acknowledgments

TBD

#### 7. References

##### 7.1. Normative References

- [I-D.ietf-nvo3-geneve]  
Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-16 (work in progress), March 2020.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4385] Bryant, S., Swallow, G., Martini, L., and D. McPherson, "Pseudowire Emulation Edge-to-Edge (PWE3) Control Word for Use over an MPLS PSN", RFC 4385, DOI 10.17487/RFC4385, February 2006, <<https://www.rfc-editor.org/info/rfc4385>>.
- [RFC5586] Bocci, M., Ed., Vigoureux, M., Ed., and S. Bryant, Ed., "MPLS Generic Associated Channel", RFC 5586, DOI 10.17487/RFC5586, June 2009, <<https://www.rfc-editor.org/info/rfc5586>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

## 7.2. Informative References

- [RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, DOI 10.17487/RFC0792, September 1981, <<https://www.rfc-editor.org/info/rfc792>>.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <<https://www.rfc-editor.org/info/rfc4291>>.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <<https://www.rfc-editor.org/info/rfc4443>>.
- [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/info/rfc4884>>.
- [RFC5082] Gill, V., Heasley, J., Meyer, D., Savola, P., Ed., and C. Pignataro, "The Generalized TTL Security Mechanism (GTSM)", RFC 5082, DOI 10.17487/RFC5082, October 2007, <<https://www.rfc-editor.org/info/rfc5082>>.
- [RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <<https://www.rfc-editor.org/info/rfc5880>>.
- [RFC7799] Morton, A., "Active and Passive Metrics and Methods (with Hybrid Types In-Between)", RFC 7799, DOI 10.17487/RFC7799, May 2016, <<https://www.rfc-editor.org/info/rfc7799>>.
- [RFC8014] Black, D., Hudson, J., Kreeger, L., Lasserre, M., and T. Narten, "An Architecture for Data-Center Network Virtualization over Layer 3 (NVO3)", RFC 8014, DOI 10.17487/RFC8014, December 2016, <<https://www.rfc-editor.org/info/rfc8014>>.
- [RFC8029] Kompella, K., Swallow, G., Pignataro, C., Ed., Kumar, N., Aldrin, S., and M. Chen, "Detecting Multiprotocol Label Switched (MPLS) Data-Plane Failures", RFC 8029, DOI 10.17487/RFC8029, March 2017, <<https://www.rfc-editor.org/info/rfc8029>>.

[RFC8762] Mirsky, G., Jun, G., Nydell, H., and R. Foote, "Simple Two-Way Active Measurement Protocol", RFC 8762, DOI 10.17487/RFC8762, March 2020, <<https://www.rfc-editor.org/info/rfc8762>>.

#### Appendix A. Additional Considerations for OAM Encapsulation Method in Geneve

Several other options of OAM encapsulation were considered. Those are listed in the Appendix solely for the informational purpose.

A Protocol Type field might be used to demultiplex active OAM protocols directly. Such method avoids the use of additional intermediate header but requires that each active OAM protocol be assigned unique identifier from the Ether Types registry maintained by IANA.

The alternative to using the Protocol Type directly is to use a shim that, in turn, identifies the OAM Protocol and, optionally, includes additional information. [RFC5586] defines how the Generic Associated Channel Label (GAL) can be used to identify that the Associated Channel Header (ACH), defined in [RFC4385], immediately follows the Bottom-of-the-Stack label. Thus, the MPLS Generic Associated Channel can be identified, and protocols are demultiplexed based on the Channel Type field's value. Number of channel types, e.g., for continuity check and performance monitoring, already have been defined and are listed in IANA MPLS Generalized Associated Channel Types (including Pseudowire Associated Channel Types) registry. The value of the Protocol Type field in the Geneve header MUST be set to MPLS to use this approach. The Geneve header MUST be immediately followed by the GAL label with the S flag set to indicate that GAL is the Bottom-of-the-stack label. Then ACH MUST follow the GAL label and the value of the Channel Type identifies which of active OAM protocols being encapsulated in the packet.

#### Authors' Addresses

Greg Mirsky  
ZTE Corp.

Email: [gregimirsky@gmail.com](mailto:gregimirsky@gmail.com)

Sami Boutros  
Ciena

Email: [sboutros@ciena.com](mailto:sboutros@ciena.com)

David Black  
Dell EMC  
176 South Street  
Hopkinton, MA 01748  
United States of America

Email: david.black@dell.com

Santosh Pallagatti  
VMware

Email: santosh.pallagatti@gmail.com

INTERNET-DRAFT  
Intended Status: Standards Track

B. Liu, Editor  
Huawei  
R. Chen  
ZTE  
F. Qin  
China Mobile  
July 8, 2019

Expires: January 9, 2020

Base YANG Data Model for NVO3 Protocols  
draft-zhang-nvo3-yang-cfg-06.txt

Abstract

This document describes the base YANG data model that can be used by operators to configure and manage Network Virtualization Overlay protocols. The model is focused on the common configuration requirement of various encapsulation options, such as VXLAN, NVGRE, GENEVE and VXLAN-GPE. Using this model as a starting point, incremental work can be done to satisfy the requirement of a specific encapsulation.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at  
<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at  
<http://www.ietf.org/shadow.html>

Copyright and License Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|  |    |
|--|----|
| 1. Introduction . . . . .                      | 3  |
| 2. Acronyms and Terminology . . . . .          | 3  |
| 2.1. Acronyms . . . . .                        | 3  |
| 2.2. Terminology . . . . .                     | 3  |
| 3. The YANG Data Model for NVO3 . . . . .      | 3  |
| 3.1 Mapping to the NVO3 architecture . . . . . | 4  |
| 3.2. The Configuration Parameters . . . . .    | 4  |
| 3.2.1. NVE as an interface . . . . .           | 4  |
| 3.2.2. Virtual Network Instance . . . . .      | 5  |
| 3.2.3. BUM Mode . . . . .                      | 5  |
| 3.3. Statistics . . . . .                      | 5  |
| 3.3. Model Structure . . . . .                 | 5  |
| 3.4. YANG Module . . . . .                     | 8  |
| 4. Security Considerations . . . . .           | 24 |
| 5. IANA Considerations . . . . .               | 24 |
| 6. Contributors . . . . .                      | 24 |
| 7. Acknowledgements . . . . .                  | 25 |
| 8. References . . . . .                        | 25 |
| 8.1. Normative References . . . . .            | 25 |
| 8.2. Informative References . . . . .          | 26 |
| Author's Addresses . . . . .                   | 27 |

## 1. Introduction

Network Virtualization Overlays (NVO3), such as VXLAN, NVGRE, GENEVE and VXLAN-GPE, enable network virtualization for data center networks environment that assumes an IP-based underlay.

YANG [RFC6020] is a data definition language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241]. This document specifies a YANG data model that can be used to configure and manage NVO3 protocols. The model covers the configuration of NVO3 instances as well as their operation states, which are the basic common requirements of the different tunnel encapsulations. Thus it is called "the base model for NVO3" in this document.

As the Network Virtualization Overlay evolves, newly defined tunnel encapsulation may require extra configuration. For example, GENEVE may require configuration of TLVs at the NVE. The base module can be augmented to accommodate these new solutions.

## 2. Acronyms and Terminology

### 2.1. Acronyms

NVO3: Network Virtualization Overlays  
VNI: Virtual Network Instance  
BUM: Broadcast, Unknown Unicast, Multicast traffic

### 2.2. Terminology

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 [RFC2119].

Familiarity with [RFC7348], [RFC7364], [RFC7365] and [RFC8014] is assumed in this document.

## 3. The YANG Data Model for NVO3

The NVO3 base YANG model defined in this document is used to configure the NVEs. It is divided into three containers. The first container contains the configuration of the virtual network instances, e.g. the VNI, the NVE that the instance is mounted, the peer NVEs which can be determined dynamically via a control plane or given statically, and the statistical states of the instance. The other two containers are separately the statistical states of the



peer NVEs and the tunnels.

### 3.1 Mapping to the NVO3 architecture

The NVO3 base YANG model is defined according to the NVO3 architecture [RFC8014]. As shown in Figure 3.1, the reference model of the NVE defined in [RFC8014], multiple instances can be mounted under a NVE. The key of the instance is VNI. The source NVE of the instance is the NVE configured by the base YANG. An instance can have several peer NVEs. A NVO3 tunnel can be determined by the VNI, the source NVE and the peer NVE. The tunnel can be built statically by manually indicate the addresses of the peer NVEs, or dynamically via a control plane, e.g. EVPN [RFC8365]. An enabler is defined in the NVO3 base YANG to choose from these two modes.

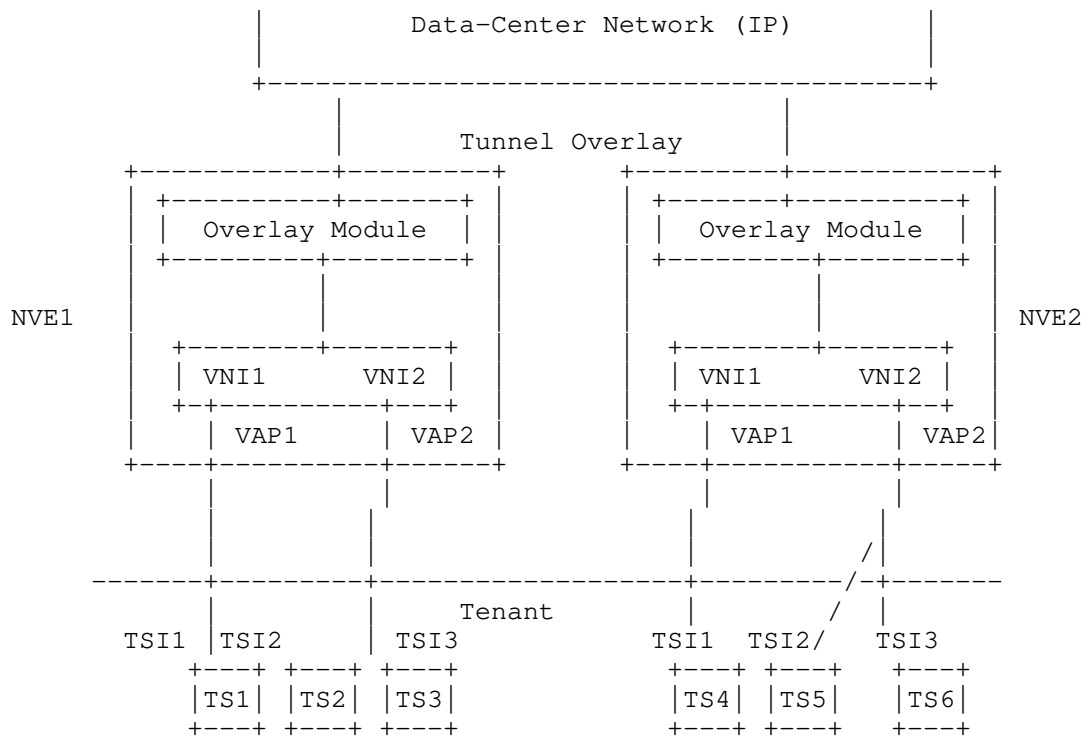


Figure 3.1. NVE Reference model in RFC 8014

### 3.2. The Configuration Parameters

#### 3.2.1. NVE as an interface

A NVE in the NVO3 base YANG is defined via augmenting the IETF

interface YANG. If anycast gateway is enabled, the source VTEP address is the address of the anycast gateway, and a bypass address is used to uniquely identify the NVE. Otherwise, the source VTEP address is the NVE interface's own IP address.

### 3.2.2. Virtual Network Instance

A Virtual Network Instance ('VNI') is a specific VN instance on an NVE [RFC7365]. At each NVE, a Tenant System is connect to VNIs through Virtual Access Points (VAP). VAPs can be physical ports or virtual ports identified by the bridge domain Identifier ('bdId'). The mapping between VNI and bdId is managed by the operator.

As defined in [draft-ietf-bess-evpn-inter-subnet-forwarding], a tenant can have multiple bridge domains, and each domain has its own VNI. Thus these VNIs are used as L2VPN. Besides, a dedicated VNI can be used for routing between the bridge domains, i.e. used as L3VPN. The mapping relationship between VNI and L2VPN (respectively, L3VPN) is given by augmenting the IETF YANG of L2VPN (respectively L3VPN).

### 3.2.3. BUM Mode

An NVE SHOULD support either ingress replication, or multicast proxy, or point to multipoint tunnels on a per-VNI basis. It is possible that both modes be used simultaneously in one NVO3 network by different NVEs.

If ingress replication is used, the receiver addresses are listed in 'peers'. If multicast proxy [RFC8293] is used, the proxy's address is given in "flood-proxy". If the choice is point to multipoint tunnels, the multicast address is given as 'multiAddr'.

### 3.3. Statistics

Operators can determine whether a NVE should gather statistic values on a per-VNI basis. An enabler is contained in the 'static' list as 'statistic-enable' leaf. If the gathering for a VNI is enabled, the statistical information about the local NVEs, the remote NVEs, the flows and the MAC addresses will be collected by the NVEs in this VNI.

### 3.3. Model Structure

```
module: ietf-nvo3
  +--rw nvo3
  |   +--rw vni-instances
  |   |   +--rw vni-instance* [vni-id]
  |   |   |   +--rw vni-id                uint32
  |   |   |   +--rw vni-mode              enumeration
```

```

+--rw source-nve                    if:interface-ref
+--rw protocol-bgp?                 boolean
+--ro status?                       vni-status-type
+--rw static-ipv4-peers
|   +--rw static-peer* [peer-ip]
|   |   +--rw peer-ip                inet:ipv4-address-no-zone
|   |   +--rw out-vni-id?            uint32
+--rw static-ipv6-peers
|   +--rw static-ipv6-peer* [peer-ip]
|   |   +--rw peer-ip                inet:ipv6-address-no-zone
+--rw flood-proxys
|   +--rw flood-proxy* [peer-ip]
|   |   +--rw peer-ip                inet:ipv4-address-no-zone
+--rw mcast-groups
|   +--rw mcast-group* [mcast-ip]
|   |   +--rw mcast-ip               inet:ipv4-address-no-zone
+--rw statistic
|   +--rw statistic-enable?          boolean
|   +--ro statistic-info
|   |   +--ro rx-bits-per-sec?        uint64
|   |   +--ro rx-pkt-per-sec?         uint64
|   |   +--ro tx-bits-per-sec?        uint64
|   |   +--ro tx-pkt-per-sec?         uint64
|   |   +--ro rx-pkts?                uint64
|   |   +--ro rx-bytes?               uint64
|   |   +--ro tx-pkts?                uint64
|   |   +--ro tx-bytes?               uint64
|   |   +--ro rx-unicast-pkts?        uint64
|   |   +--ro rx-multicast-pkts?      uint64
|   |   +--ro rx-broadcast-pkts?      uint64
|   |   +--ro drop-unicast-pkts?      uint64
|   |   +--ro drop-multicast-pkts?    uint64
|   |   +--ro drop-broadcast-pkts?    uint64
|   |   +--ro tx-unicast-pkts?        uint64
|   |   +--ro tx-multicast-pkts?      uint64
|   |   +--ro tx-broadcast-pkts?      uint64
+--ro vni-peer-infos
|   +--ro peers
|   |   +--ro peer* [vni-id source-ip peer-ip]
|   |   |   +--ro vni-id              uint32
|   |   |   +--ro source-ip           inet:ip-address-no-zone
|   |   |   +--ro peer-ip             inet:ip-address-no-zone
|   |   |   +--ro tunnel-type?        peer-type
|   |   |   +--ro out-vni-id?         uint32
+--ro tunnel-infos
|   +--ro tunnel-info* [tunnel-id]
|   |   +--ro tunnel-id               uint32
|   |   +--ro source-ip?              inet:ip-address-no-zone

```

```

    +--ro peer-ip?          inet:ip-address-no-zone
    +--ro status?           tunnel-status
    +--ro type?             tunnel-type
    +--ro up-time?          string
    +--ro vrf-name?         -> /ni:network-instances/network-instance/name

augment /if:interfaces/if:interface:
  +--rw nvo3-nve
    +--rw nvo3-config
      +--rw source-vtep-ip?    inet:ipv4-address-no-zone
      +--rw source-vtep-ipv6?  inet:ipv6-address-no-zone
      +--rw bypass-vtep-ip?    inet:ipv4-address-no-zone
      +--rw statistics
        +--rw statistic* [vni-id mode peer-ip direction]
          +--rw vni-id        uint32
          +--rw mode           vni-type
          +--rw peer-ip        inet:ipv4-address-no-zone
          +--rw direction      direction-type
          +--ro info
            +--ro rx-pkts?      uint64
            +--ro rx-bytes?     uint64
            +--ro tx-pkts?      uint64
            +--ro tx-bytes?     uint64
            +--ro rx-unicast-pkts? uint64
            +--ro rx-multicast-pkts? uint64
            +--ro rx-broadcast-pkts? uint64
            +--ro tx-unicast-pkts? uint64
            +--ro tx-multicast-pkts? uint64
            +--ro tx-broadcast-pkts? uint64
            +--ro drop-unicast-pkts? uint64
            +--ro drop-multicast-pkts? uint64
            +--ro drop-broadcast-pkts? uint64
            +--ro rx-bits-per-sec? uint64
            +--ro rx-pkt-per-sec? uint64
            +--ro tx-bits-per-sec? uint64
            +--ro tx-pkt-per-sec? uint64
      +--rw nvo3-gateway
        +--rw nvo3-gateway
          +--rw vxlan-anycast-gateway? boolean
    augment /ni:network-instances/ni:network-instance/ni:ni-type/l3vpn:l3vpn/l3
    vpn:l3vpn:
      +--rw vni-lists
        +--rw vni* [vni-id]
          +--rw vni-id        uint32
    augment /ni:network-instances/ni:network-instance/ni:ni-type/l2vpn:l2vpn:
    +--rw vni-lists
      +--rw vni* [vni-id]
        +--rw vni-id          uint32
        +--rw split-horizon-mode? vni-bind-type

```

```

        +---rw split-group?          string

    rpcs:
      +---x reset-vni-instance-statistic
      |   +---w input
      |       +---w vni-id          uint32
      +---x reset-vni-peer-statistic
      |   +---w input
      |       +---w vni-id          uint32
      |       +---w mode            vni-type
      |       +---w peer-ip         inet:ipv4-address-no-zone
      |       +---w direction       direction-type

```

Figure 3.2. The tree structure of YANG module for NVO3 configuration

### 3.4. YANG Module

```

<CODE BEGINS> file "ietf-nvo3-base@2019-07-01.yang"
module ietf-nvo3 {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-nvo3";
  prefix "nvo3";

  import ietf-network-instance {
    prefix "ni";
  }

  import ietf-interfaces {
    prefix "if";
  }

  import ietf-inet-types {
    prefix "inet";
  }

  import ietf-l2vpn {
    prefix "l2vpn";
  }

  import ietf-bgp-l3vpn {
    prefix "l3vpn";
  }

  organization "ietf";
  contact "ietf";
  description "Yang model for NVO3";

  revision 2019-04-01 {

```

```
    description
      "Init revision";
    reference
      "";
  }

  typedef vni-status-type {
    type enumeration {
      enum "up" {
        description
          "Vni status up.";
      }
      enum "down" {
        description
          "Vni status down.";
      }
    }
    description
      "Vni status";
  }

  typedef vni-type {
    type enumeration {
      enum "l2" {
        description
          "layer 2 mode";
      }
      enum "l3" {
        description
          "layer 3 mode";
      }
    }
    description
      "vni type";
  }

  typedef peer-type {
    type enumeration {
      enum "static" {
        description
          "Static.";
      }
      enum "dynamic" {
        description
          "Dynamic.";
      }
    }
    description
```

```
        "Peer type";
    }

    typedef tunnel-status {
        type enumeration {
            enum "up" {
                description
                    "The tunnel is up.";
            }
            enum "down" {
                description
                    "The tunnel is down.";
            }
        }
        description
            "Tunnel status";
    }

    typedef tunnel-type {
        type enumeration {
            enum "dynamic" {
                description
                    "The tunnel is dynamic.";
            }
            enum "static" {
                description
                    "The tunnel is static.";
            }
            enum "invalid" {
                description
                    "The tunnel is invalid.";
            }
        }
        description
            "Tunnel type";
    }

    typedef direction-type {
        type enumeration {
            enum "inbound" {
                description
                    "Inbound.";
            }
            enum "outbound" {
                description
                    "Outbound.";
            }
            enum "bidirection" {
                description
```

```
        "Bidirection.";
    }
}
description
    "Bound direction";
}

typedef vni-bind-type {
    type enumeration {
        enum "hub-mode" {
            description
                "Hub mode.";
        }
        enum "spoke-mode" {
            description
                "Spoke mode.";
        }
    }
    description
        "bdBindVniType";
}

container nvo3 {
    description
        "Management of NVO3.";

    container vni-instances {
        description
            "The confiuration and information table of the VNI.";
        list vni-instance {
            key "vni-id";
            must "(/if:interfaces/if:interface[if:name = current()/source_nve]/if
:type = 'Nve')";
            description
                "The confiuration and information of the VNI.";
            leaf vni-id {
                type uint32 {
                    range "1..16777215";
                }
                description
                    "The id of VNI.";
            }
            leaf vni-mode {
                type enumeration {
                    enum "Local" {
                        description
                            "Local mode";
                    }
                    enum "Global" {
```



```
        description
            "Global mode";
        }
    }
    description
        "The mode of the VNI instance.";
}
leaf source-nve {
    type if:interface-ref;
    mandatory true;
    description
        "The name of the nve interface .";
}
leaf protocol-bgp {
    type boolean;
    default "false";
    description
        "Whether use bgp as vxlan's protocol.";
}
leaf status {
    type vni-status-type;
    config false;
    description
        "The status of the VNI.";
}
container static-ipv4-peers {
    description
        "The remote NVE address table in a same VNI.";
    list static-peer {
        key "peer-ip";
        description
            "The remote NVE address in a same VNI.";
        leaf peer-ip {
            type inet:ipv4-address-no-zone;
            description
                "The address of the NVE.";
        }
        leaf out-vni-id {
            type uint32 {
                range "1..16777215";
            }
            description
                "The ID of the out VNI. Do not support separate deletion.";
        }
    }
}
container static-ipv6-peers {
    description
```

```
    "The remote NVE ipv6 address table in a same VNI.";
list static-ipv6-peer {
    key "peer-ip";
    description
        "The remote NVE ipv6 address in a same VNI.";
    leaf peer-ip {
        type inet:ipv6-address-no-zone;
        description
            "The ipv6 address of the NVE.";
    }
}
}
container flood-proxys {
    description
        "The flood proxys for this VNI";
    list flood-proxy {
        key "peer-ip";
        leaf peer-ip {
            type inet:ipv4-address-no-zone;
            description
                "peer ip address";
        }
        description
            "List of the flood proxys";
    }
}
container mcast-groups {
    description
        "The mcast address table.";
    list mcast-group {
        key "mcast-ip";
        description
            "The mcast address.";
        leaf mcast-ip {
            type inet:ipv4-address-no-zone;
            description
                "The mcast address of NVO3.";
        }
    }
}
}
container statistic {
    description
        "The VNI member in a same NVE.";
    leaf statistic-enable {
        type boolean;
        default "false";
        description
            "To determine whether to enable the statistics for a VNI.";
```

```
}
container statistic-info {
  config false;
  description
    "The vni instance traffic statistics information.";
  leaf rx-bits-per-sec {
    type uint64;
    config false;
    description
      "Number of bits received per second.";
  }
  leaf rx-pkt-per-sec {
    type uint64;
    config false;
    description
      "Number of packets received per second.";
  }
  leaf tx-bits-per-sec {
    type uint64;
    config false;
    description
      "Number of bits sent per second.";
  }
  leaf tx-pkt-per-sec {
    type uint64;
    config false;
    description
      "Number of packets sent per second.";
  }
  leaf rx-pkts {
    type uint64;
    config false;
    description
      "Total number of received packets.";
  }
  leaf rx-bytes {
    type uint64;
    config false;
    description
      "Total number of received bytes.";
  }
  leaf tx-pkts {
    type uint64;
    config false;
    description
      "Total number of sent packets.";
  }
  leaf tx-bytes {
```

```
    type uint64;
    config false;
    description
        "Total number of sent bytes.";
}
leaf rx-unicast-pkts {
    type uint64;
    config false;
    description
        "Number of received unicast packets.";
}
leaf rx-multicast-pkts {
    type uint64;
    config false;
    description
        "Number of received multicast packets.";
}
leaf rx-broadcast-pkts {
    type uint64;
    config false;
    description
        "Number of received broadcast packets.";
}
leaf drop-unicast-pkts {
    type uint64;
    config false;
    description
        "Number of discarded unicast packets.";
}
leaf drop-multicast-pkts {
    type uint64;
    config false;
    description
        "Number of discarded multicast packets.";
}
leaf drop-broadcast-pkts {
    type uint64;
    config false;
    description
        "Number of discarded broadcast packets.";
}
leaf tx-unicast-pkts {
    type uint64;
    config false;
    description
        "Number of sent unicast packets.";
}
leaf tx-multicast-pkts {
```

```

        type uint64;
        config false;
        description
            "Number of sent multicast packets.";
    }
    leaf tx-broadcast-pkts {
        type uint64;
        config false;
        description
            "Number of sent broadcast packets.";
    }
}

}

}

}

container vni-peer-infos {
    config false;
    description
        "The information table of vni members.";
    container peers {
        config false;
        description
            "The remote nve address in a same VNI.";
        list peer {
            key "vni-id source-ip peer-ip";
            config false;
            description
                "The remote nve address list in a same VNI.";
            leaf vni-id {
                type uint32 {
                    range "1..16777215";
                }
                config false;
                description
                    "The ID of VNI.";
            }
            leaf source-ip {
                type inet:ip-address-no-zone;
                config false;
                description
                    "The source address of the NVE interface.";
            }
            leaf peer-ip {
                type inet:ip-address-no-zone;
                config false;
                description

```

```
        "The remote NVE address.";
    }
    leaf tunnel-type {
        type peer-type;
        config false;
        description
            "Tunnel type.";
    }
    leaf out-vni-id {
        type uint32 {
            range "1..16777215";
        }
        config false;
        description
            "The ID of the out VNI.";
    }
}
}
}

container tunnel-infos {
    config false;
    description
        "VxLAN tunnel information.";
    list tunnel-info {
        key "tunnel-id";
        config false;
        description
            "VxLAN tunnel information list.";
        leaf tunnel-id {
            type uint32 {
                range "1..4294967295";
            }
            config false;
            description
                "The ID of Vxlan tunnel.";
        }
        leaf source-ip {
            type inet:ip-address-no-zone;
            config false;
            description
                "Local NVE interface address.";
        }
        leaf peer-ip {
            type inet:ip-address-no-zone;
            config false;
            description
                "Remote NVE interface address.";
```

```
    }
    leaf status {
        type tunnel-status;
        config false;
        description
            "Tunnel status.";
    }
    leaf type {
        type tunnel-type;
        config false;
        description
            "Tunnel type.";
    }
    leaf up-time {
        type string {
            length "1..10";
        }
        config false;
        description
            "Vxlan tunnel up time.";
    }
    leaf vrf-name {
        type leafref {
            path "/ni:network-instances/ni:network-instance/ni:name";
        }
        default "_public_";
        config false;
        description
            "The name of VPN instance.";
    }
}

augment "/if:interfaces/if:interface" {
    description
        "Augment the interface, NVE as an interface.";
    container nvo3-nve {
        when "if:interfaces/if:interface/if:type = 'Nve'";
        description
            "Network virtualization edge.";
        leaf source-vtep-ip {
            type inet:ipv4-address-no-zone;
            description
                "The source address of the NVE interface.";
        }
        leaf source-vtep-ipv6 {
            type inet:ipv6-address-no-zone;
            description
                "The source address of the NVE interface.";
```

```
        "The source ipv6 address of the NVE interface.";
    }
    leaf bypass-vtep-ip {
        type inet:ipv4-address-no-zone;
        description
            "The source address of bypass VXLAN tunnel.";
    }
    container statistics {
        description
            "VXLAN Tunnel Traffic Statistical Configuration Table.";
        list statistic {
            key "vni-id mode peer-ip direction";
            description
                "VXLAN Tunnel Traffic Statistics Configuration.";
            leaf vni-id {
                type uint32 {
                    range "1..16777215";
                }
                description
                    "ID of the VNI.";
            }
            leaf mode {
                type vni-type;
                description
                    "The type of the NVE interface.";
            }
            leaf peer-ip {
                type inet:ipv4-address-no-zone;
                description
                    "IP address of the remote VTEP.";
            }
            leaf direction {
                type direction-type;
                must "(./mode='l3' and ./bound!='bidirection')";
                description
                    "Traffic statistics type about the VXLAN tunnel.";
            }
        }
        container info {
            config false;
            description
                "Traffic statistics about the peer.";
            leaf rx-pkts {
                type uint64;
                config false;
                description
                    "Total number of received packets.";
            }
            leaf rx-bytes {
```



```
        type uint64;
        config false;
        description
            "Total number of received bytes.";
    }
    leaf tx-pkts {
        type uint64;
        config false;
        description
            "Total number of sent packets.";
    }
    leaf tx-bytes {
        type uint64;
        config false;
        description
            "Total number of sent bytes.";
    }
    leaf rx-unicast-pkts {
        type uint64;
        config false;
        description
            "Number of received unicast packets.";
    }
    leaf rx-multicast-pkts {
        type uint64;
        config false;
        description
            "Number of received multicast packets.";
    }
    leaf rx-broadcast-pkts {
        type uint64;
        config false;
        description
            "Number of received broadcast packets.";
    }
    leaf tx-unicast-pkts {
        type uint64;
        config false;
        description
            "Number of sent unicast packets.";
    }
    leaf tx-multicast-pkts {
        type uint64;
        config false;
        description
            "Number of sent multicast packets.";
    }
    leaf tx-broadcast-pkts {
```

```
        type uint64;
        config false;
        description
            "Number of sent broadcast packets.";
    }
    leaf drop-unicast-pkts {
        type uint64;
        config false;
        description
            "Number of discarded unicast packets.";
    }
    leaf drop-multicast-pkts {
        type uint64;
        config false;
        description
            "Number of discarded multicast packets.";
    }
    leaf drop-broadcast-pkts {
        type uint64;
        config false;
        description
            "Number of discarded broadcast packets.";
    }
    leaf rx-bits-per-sec {
        type uint64;
        config false;
        description
            "Number of bits received per second.";
    }
    leaf rx-pkt-per-sec {
        type uint64;
        config false;
        description
            "Number of packets received per second.";
    }
    leaf tx-bits-per-sec {
        type uint64;
        config false;
        description
            "Number of bits sent per second.";
    }
    leaf tx-pkt-per-sec {
        type uint64;
        config false;
        description
            "Number of packets sent per second.";
    }
}
```

```
    }
  }

}

container nvo3-gateway {
  when "if:interfaces/if:interface/if:type = 'Vbdif'";
  description
    "Enable anycast gateway.";
  leaf vxlan-anycast-gateway {
    type boolean;
    default "false";
    description
      "Enable vxlan anycast gateway.";
  }
}

}

augment "/ni:network-instances/ni:network-instance/ni:ni-type" +
  "/l3vpn:l3vpn/l3vpn:l3vpn" {
  description "Augment for l3vpn instance";
  container vni-lists {
    description "Vni list for l3vpn";
    list vni {
      key "vni-id";
      description
        "Vni for current l3vpn instance";
      leaf vni-id {
        type uint32 {
          range "1..16777215";
        }
        description
          "The id of VNI.";
      }
    }
  }
}

}

augment "/ni:network-instances/ni:network-instance/ni:ni-type" +
  "/l2vpn:l2vpn" {
  description "Augment for l2vpn instance";
  container vni-lists {
    description "Vni list for l2vpn";
    list vni {
      key "vni-id";
      description
        "Vni for current l2vpn instance";
      leaf vni-id {
        type uint32 {
```

```
        range "1..16777215";
    }
    description
        "The id of VNI.";
}
leaf split-horizon-mode {
    type vni-bind-type;
    default "hub-mode";
    description
        "Split horizon mode.";
}
leaf split-group {
    type string {
        length "1..31";
    }
    description
        "Split group name.";
}
}
}
}

rpc reset-vni-instance-statistic {
    description
        "Clear traffic statistics about the VNI.";
    input {
        leaf vni-id {
            type uint32 {
                range "1..16777215";
            }
            mandatory true;
            description
                "ID of the VNI.";
        }
    }
}

rpc reset-vni-peer-statistic {
    description
        "Clear traffic statistics about the VXLAN tunnel.";
    input {
        leaf vni-id {
            type uint32 {
                range "1..16777215";
            }
            mandatory true;
            description
                "ID of the VNI.";
        }
    }
}
```

```
    leaf mode {
        type vni-type;
        mandatory true;
        description
            "The type of vni memeber statistic.";
    }
    leaf peer-ip {
        type inet:ipv4-address-no-zone;
        mandatory true;
        description
            "IP address of the remote NVE interface.";
    }
    leaf direction{
        type direction-type;
        must "(./mode='mode-l3' and ./bound!='bidirection')";
        mandatory true;
        description
            "Traffic statistics type about the VXLAN tunnel.";
    }
}
}
```

<CODE ENDS>

#### 4. Security Considerations

This document raises no new security issues.

#### 5. IANA Considerations

The namespace URI defined in Section 3.3 need be registered in the IETF XML registry [RFC3688].

This document need to register the 'ietf-nvo3-base' YANG module in the YANG Module Names registry [RFC6020].

#### 6. Contributors

Haibo Wang  
Huawei  
Email: rainsword.wang@huawei.com

Yuan Gao  
Huawei  
Email: sean.gao@huawei.com

Gang Yan

Huawei  
Email: yangang@huawei.com

Mingui Zhang  
Huawei  
Email: zhangmingui@huawei.com

Yubao (Bob) Wang  
ZTE Corporation  
Email: yubao.wang2008@hotmail.com

Ruixue Wang  
China Mobile  
Email: wangruixue@chinamobile.com

Sijun Weng  
China Mobile  
Email: wengsijun@chinamobile.com

## 7. Acknowledgements

Authors would like to thank the comments and suggestions from Tao Han, Weilian Jiang.

## 8. References

### 8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC7364] T. Narten, E. Gray, et al, "Problem Statement: Overlays for Network Virtualization", draft-ietf-nvo3-overlay-problem-statement, working in progress.
- [RFC7365] Marc Lasserre, Florin Balus, et al, "Framework for DC Network Virtualization", draft-ietf-nvo3-framework, working in progress.
- [RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, August 2014.
- [I-D.ietf-nvo3-geneve] Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-

nvo3-geneve-10 (work in progress), March 2019.

- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.
- [RFC8014] D. Black, J. Hudson, L. Kreeger, M. Lasserre, T. Narten, An Architecture for Data-Center Network Virtualization over Layer 3 (NVO3), RFC8014, December 2016.

## 8.2. Informative References

- [RFC7637] M. Sridharan, A. Greenberg, et al, "NVGRE: Network Virtualization using Generic Routing Encapsulation", RFC7637, September 2015.
- [I-D.ietf-nvo3-vxlan-gpe] Maino, F., Kreeger, L., and U. Elzur, "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-06 (work in progress), April 2018.
- [I-D.draft-ietf-bess-evpn-inter-subnet-forwarding] A. Sajassi, S. Salam, S. Thoria, J. Drake, J. Rabadan, "Integrated Routing and Bridging in EVPN", draft-ietf-bess-evpn-inter-subnet-forwarding-08, March 4, 2019.
- [RFC8293] A. Ghanwani, L. Dunbar, V. Bannai, M. McBride, R. Krishnan, "A Framework for Multicast in Network Virtualization over Layer 3", RFC8293, January 2018.

## Author's Addresses

Bing Liu  
Huawei Technologies  
No. 156 Beiqing Rd. Haidian District,  
Beijing 100095  
P.R. China

Email: [remy.liubing@huawei.com](mailto:remy.liubing@huawei.com)

Ran Chen  
ZTE Corporation

Email: [chen.ran@zte.com.cn](mailto:chen.ran@zte.com.cn)

Fengwei Qin  
China Mobile  
32 Xuanwumen West Ave, Xicheng District  
Beijing, Beijing 100053  
China

Email: [qinfengwei@chinamobile.com](mailto:qinfengwei@chinamobile.com)



INTERNET-DRAFT  
Intended Status: Standards Track

B. Liu, Ed.  
Huawei  
R. Chen  
ZTE  
F. Qin  
China Mobile  
R. Rahman  
Cisco

Expires: March 9, 2020

September 6, 2019

Base YANG Data Model for NVO3 Protocols  
draft-zhang-nvo3-yang-cfg-07.txt

Abstract

This document describes the base YANG data model that can be used by operators to configure and manage Network Virtualization Overlay protocols. The model is focused on the common configuration requirement of various encapsulation options, such as VXLAN, NVGRE, GENEVE and VXLAN-GPE. Using this model as a starting point, incremental work can be done to satisfy the requirement of a specific encapsulation.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at  
<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at  
<http://www.ietf.org/shadow.html>

Copyright and License Notice

Copyright (c) 2019 IETF Trust and the persons identified as the

document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|  |    |
|--|----|
| 1. Introduction . . . . .                      | 3  |
| 2. Acronyms and Terminology . . . . .          | 3  |
| 2.1. Acronyms . . . . .                        | 3  |
| 2.2. Terminology . . . . .                     | 3  |
| 3. The YANG Data Model for NVO3 . . . . .      | 3  |
| 3.1 Mapping to the NVO3 architecture . . . . . | 4  |
| 3.2. The Configuration Parameters . . . . .    | 4  |
| 3.2.1. NVE as an interface . . . . .           | 4  |
| 3.2.2. Virtual Network Instance . . . . .      | 5  |
| 3.2.3. BUM Mode . . . . .                      | 5  |
| 3.3. Statistics . . . . .                      | 5  |
| 3.3. Model Structure . . . . .                 | 5  |
| 3.4. YANG Module . . . . .                     | 8  |
| 4. Security Considerations . . . . .           | 24 |
| 5. IANA Considerations . . . . .               | 24 |
| 6. Contributors . . . . .                      | 24 |
| 7. Acknowledgements . . . . .                  | 25 |
| 8. References . . . . .                        | 25 |
| 8.1. Normative References . . . . .            | 25 |
| 8.2. Informative References . . . . .          | 26 |
| Author's Addresses . . . . .                   | 27 |

## 1. Introduction

Network Virtualization Overlays (NVO3), such as VXLAN, NVGRE, GENEVE and VXLAN-GPE, enable network virtualization for data center networks environment that assumes an IP-based underlay.

YANG [RFC6020] is a data definition language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241]. This document specifies a YANG data model that can be used to configure and manage NVO3 protocols. The model covers the configuration of NVO3 instances as well as their operation states, which are the basic common requirements of the different tunnel encapsulations. Thus it is called "the base model for NVO3" in this document.

As the Network Virtualization Overlay evolves, newly defined tunnel encapsulation may require extra configuration. For example, GENEVE may require configuration of TLVs at the NVE. The base module can be augmented to accommodate these new solutions.

## 2. Acronyms and Terminology

### 2.1. Acronyms

NVO3: Network Virtualization Overlays  
VNI: Virtual Network Instance  
BUM: Broadcast, Unknown Unicast, Multicast traffic

### 2.2. Terminology

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 [RFC2119].

Familiarity with [RFC7348], [RFC7364], [RFC7365] and [RFC8014] is assumed in this document.

## 3. The YANG Data Model for NVO3

The NVO3 base YANG model defined in this document is used to configure the NVEs. It is divided into three containers. The first container contains the configuration of the virtual network instances, e.g. the VNI, the NVE that the instance is mounted, the peer NVEs which can be determined dynamically via a control plane or given statically, and the statistical states of the instance. The other two containers are separately the statistical states of the

peer NVEs and the tunnels.

### 3.1 Mapping to the NVO3 architecture

The NVO3 base YANG model is defined according to the NVO3 architecture [RFC8014]. As shown in Figure 3.1, the reference model of the NVE defined in [RFC8014], multiple instances can be mounted under a NVE. The key of the instance is VNI. The source NVE of the instance is the NVE configured by the base YANG. An instance can have several peer NVEs. A NVO3 tunnel can be determined by the VNI, the source NVE and the peer NVE. The tunnel can be built statically by manually indicate the addresses of the peer NVEs, or dynamically via a control plane, e.g. EVPN [RFC8365]. An enabler is defined in the NVO3 base YANG to choose from these two modes.

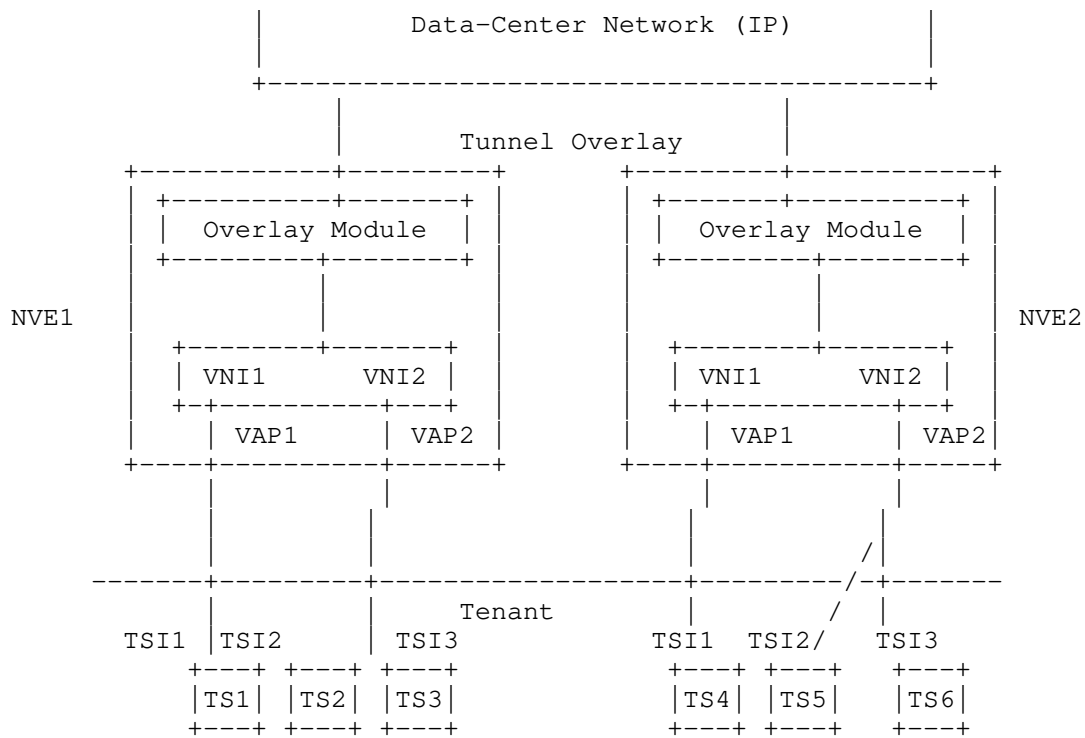


Figure 3.1. NVE Reference model in RFC 8014

### 3.2. The Configuration Parameters

#### 3.2.1. NVE as an interface

A NVE in the NVO3 base YANG is defined via augmenting the IETF

### 3.2.2. Virtual Network Instance

As defined in [draft-ietf-bess-evpn-inter-subnet-forwarding], a tenant can have multiple bridge domains, and each domain has its own VNI. Thus these VNIs are used as L2VPN. Besides, a dedicated VNI can be used for routing between the bridge domains, i.e. used as L3VPN. The mapping relationship between VNI and L2VPN (respectively, L3VPN) is given by augmenting the IETF YANG of L2VPN (respectively L3VPN).

An NVE SHOULD support either ingress replication, or multicast proxy, or point to multipoint tunnels on a per-VNI basis. It is possible that both modes be used simultaneously in one NVO3 network by different NVEs.

### 3.3. Statistics

### 3.3. Model Structure

Bing Liu, et al

```

+--rw source-nve                    if:interface-ref
+--rw protocol-bgp?                 boolean
+--ro status?                        vni-status-type
+--rw static-ipv4-peers
|   +--rw static-peer* [peer-ip]
|   |   +--rw peer-ip                inet:ipv4-address-no-zone
|   |   +--rw out-vni-id?            uint32
+--rw static-ipv6-peers
|   +--rw static-ipv6-peer* [peer-ip]
|   |   +--rw peer-ip                inet:ipv6-address-no-zone
+--rw flood-proxys
|   +--rw flood-proxy* [peer-ip]
|   |   +--rw peer-ip                inet:ipv4-address-no-zone
+--rw mcast-groups
|   +--rw mcast-group* [mcast-ip]
|   |   +--rw mcast-ip                inet:ipv4-address-no-zone
+--rw statistic
|   +--rw statistic-enable?          boolean
|   +--ro statistic-info
|   |   +--ro rx-bits-per-sec?        uint64
|   |   +--ro rx-pkt-per-sec?         uint64
|   |   +--ro tx-bits-per-sec?        uint64
|   |   +--ro tx-pkt-per-sec?        uint64
|   |   +--ro rx-pkts?                uint64
|   |   +--ro rx-bytes?               uint64
|   |   +--ro tx-pkts?                uint64
|   |   +--ro tx-bytes?               uint64
|   |   +--ro rx-unicast-pkts?        uint64
|   |   +--ro rx-multicast-pkts?      uint64
|   |   +--ro rx-broadcast-pkts?      uint64
|   |   +--ro drop-unicast-pkts?      uint64
|   |   +--ro drop-multicast-pkts?    uint64
|   |   +--ro drop-broadcast-pkts?    uint64
|   |   +--ro tx-unicast-pkts?        uint64
|   |   +--ro tx-multicast-pkts?      uint64
|   |   +--ro tx-broadcast-pkts?      uint64
+--ro vni-peer-infos
|   +--ro peers
|   |   +--ro peer* [vni-id source-ip peer-ip]
|   |   |   +--ro vni-id              uint32
|   |   |   +--ro source-ip           inet:ip-address-no-zone
|   |   |   +--ro peer-ip             inet:ip-address-no-zone
|   |   |   +--ro tunnel-type?        peer-type
|   |   |   +--ro out-vni-id?         uint32
+--ro tunnel-infos
|   +--ro tunnel-info* [tunnel-id]
|   |   +--ro tunnel-id               uint32
|   |   +--ro source-ip?              inet:ip-address-no-zone

```

```

    +--ro peer-ip?      inet:ip-address-no-zone
    +--ro status?       tunnel-status
    +--ro type?         tunnel-type
    +--ro up-time?      string
    +--ro vrf-name?     -> /ni:network-instances/network-instance/name

augment /if:interfaces/if:interface:
  +--rw nvo3-nve
    +--rw nvo3-config
      +--rw source-vtep-ip?      inet:ipv4-address-no-zone
      +--rw source-vtep-ipv6?    inet:ipv6-address-no-zone
      +--rw bypass-vtep-ip?      inet:ipv4-address-no-zone
      +--rw statistics
        +--rw statistic* [vni-id mode peer-ip direction]
          +--rw vni-id          uint32
          +--rw mode             vni-type
          +--rw peer-ip          inet:ipv4-address-no-zone
          +--rw direction        direction-type
          +--ro info
            +--ro rx-pkts?      uint64
            +--ro rx-bytes?     uint64
            +--ro tx-pkts?      uint64
            +--ro tx-bytes?     uint64
            +--ro rx-unicast-pkts? uint64
            +--ro rx-multicast-pkts? uint64
            +--ro rx-broadcast-pkts? uint64
            +--ro tx-unicast-pkts? uint64
            +--ro tx-multicast-pkts? uint64
            +--ro tx-broadcast-pkts? uint64
            +--ro drop-unicast-pkts? uint64
            +--ro drop-multicast-pkts? uint64
            +--ro drop-broadcast-pkts? uint64
            +--ro rx-bits-per-sec? uint64
            +--ro rx-pkt-per-sec? uint64
            +--ro tx-bits-per-sec? uint64
            +--ro tx-pkt-per-sec? uint64
      +--rw nvo3-gateway
        +--rw nvo3-gateway
          +--rw vxlan-anycast-gateway? boolean
    augment /ni:network-instances/ni:network-instance/ni:ni-type/l3vpn:l3vpn/l3
    vpn:l3vpn:
      +--rw vni-lists
        +--rw vni* [vni-id]
          +--rw vni-id          uint32
    augment /ni:network-instances/ni:network-instance/ni:ni-type/l2vpn:l2vpn:
    +--rw vni-lists
      +--rw vni* [vni-id]
        +--rw vni-id          uint32
        +--rw split-horizon-mode? vni-bind-type

```

```

        +---rw split-group?          string

    rpcs:
      +---x reset-vni-instance-statistic
      |   +---w input
      |   |   +---w vni-id          uint32
      +---x reset-vni-peer-statistic
      |   +---w input
      |   |   +---w vni-id          uint32
      |   |   +---w mode            vni-type
      |   |   +---w peer-ip         inet:ipv4-address-no-zone
      |   |   +---w direction       direction-type

```

Figure 3.2. The tree structure of YANG module for NVO3 configuration

### 3.4. YANG Module

```

<CODE BEGINS> file "ietf-nvo3-base@2019-07-01.yang"
module ietf-nvo3 {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-nvo3";
  prefix "nvo3";

  import ietf-network-instance {
    prefix "ni";
  }

  import ietf-interfaces {
    prefix "if";
  }

  import ietf-inet-types {
    prefix "inet";
  }

  import ietf-l2vpn {
    prefix "l2vpn";
  }

  import ietf-bgp-l3vpn {
    prefix "l3vpn";
  }

  organization "ietf";
  contact "ietf";
  description "Yang model for NVO3";

  revision 2019-04-01 {

```



```
    description
      "Init revision";
    reference
      "";
  }

  typedef vni-status-type {
    type enumeration {
      enum "up" {
        description
          "Vni status up.";
      }
      enum "down" {
        description
          "Vni status down.";
      }
    }
    description
      "Vni status";
  }

  typedef vni-type {
    type enumeration {
      enum "l2" {
        description
          "layer 2 mode";
      }
      enum "l3" {
        description
          "layer 3 mode";
      }
    }
    description
      "vni type";
  }

  typedef peer-type {
    type enumeration {
      enum "static" {
        description
          "Static.";
      }
      enum "dynamic" {
        description
          "Dynamic.";
      }
    }
    description
```

```
        "Peer type";
    }

typedef tunnel-status {
    type enumeration {
        enum "up" {
            description
                "The tunnel is up.";
        }
        enum "down" {
            description
                "The tunnel is down.";
        }
    }
    description
        "Tunnel status";
}

typedef tunnel-type {
    type enumeration {
        enum "dynamic" {
            description
                "The tunnel is dynamic.";
        }
        enum "static" {
            description
                "The tunnel is static.";
        }
        enum "invalid" {
            description
                "The tunnel is invalid.";
        }
    }
    description
        "Tunnel type";
}

typedef direction-type {
    type enumeration {
        enum "inbound" {
            description
                "Inbound.";
        }
        enum "outbound" {
            description
                "Outbound.";
        }
        enum "bidirection" {
            description
```

```
        "Bidirection.";
    }
}
description
    "Bound direction";
}

typedef vni-bind-type {
    type enumeration {
        enum "hub-mode" {
            description
                "Hub mode.";
        }
        enum "spoke-mode" {
            description
                "Spoke mode.";
        }
    }
    description
        "bdBindVniType";
}

container nvo3 {
    description
        "Management of NVO3.";

    container vni-instances {
        description
            "The confiuration and information table of the VNI.";
        list vni-instance {
            key "vni-id";
            must "(/if:interfaces/if:interface[if:name = current()/source_nve]/if
:type = 'Nve')";
            description
                "The confiuration and information of the VNI.";
            leaf vni-id {
                type uint32 {
                    range "1..16777215";
                }
                description
                    "The id of VNI.";
            }
            leaf vni-mode {
                type enumeration {
                    enum "Local" {
                        description
                            "Local mode";
                    }
                    enum "Global" {
```

```
        description
            "Global mode";
        }
    }
    description
        "The mode of the VNI instance.";
}
leaf source-nve {
    type if:interface-ref;
    mandatory true;
    description
        "The name of the nve interface .";
}
leaf protocol-bgp {
    type boolean;
    default "false";
    description
        "Whether use bgp as vxlan's protocol.";
}
leaf status {
    type vni-status-type;
    config false;
    description
        "The status of the VNI.";
}
container static-ipv4-peers {
    description
        "The remote NVE address table in a same VNI.";
    list static-peer {
        key "peer-ip";
        description
            "The remote NVE address in a same VNI.";
        leaf peer-ip {
            type inet:ipv4-address-no-zone;
            description
                "The address of the NVE.";
        }
        leaf out-vni-id {
            type uint32 {
                range "1..16777215";
            }
            description
                "The ID of the out VNI. Do not support separate deletion.";
        }
    }
}
container static-ipv6-peers {
    description
```

```
    "The remote NVE ipv6 address table in a same VNI.";
  list static-ipv6-peer {
    key "peer-ip";
    description
      "The remote NVE ipv6 address in a same VNI.";
    leaf peer-ip {
      type inet:ipv6-address-no-zone;
      description
        "The ipv6 address of the NVE.";
    }
  }
}
container flood-proxys {
  description
    "The flood proxys for this VNI";
  list flood-proxy {
    key "peer-ip";
    leaf peer-ip {
      type inet:ipv4-address-no-zone;
      description
        "peer ip address";
    }
    description
      "List of the flood proxys";
  }
}
container mcast-groups {
  description
    "The mcast address table.";
  list mcast-group {
    key "mcast-ip";
    description
      "The mcast address.";
    leaf mcast-ip {
      type inet:ipv4-address-no-zone;
      description
        "The mcast address of NVO3.";
    }
  }
}
container statistic {
  description
    "The VNI member in a same NVE.";
  leaf statistic-enable {
    type boolean;
    default "false";
    description
      "To determine whether to enable the statistics for a VNI.";
```

```
}
container statistic-info {
  config false;
  description
    "The vni instance traffic statistics information.";
  leaf rx-bits-per-sec {
    type uint64;
    config false;
    description
      "Number of bits received per second.";
  }
  leaf rx-pkt-per-sec {
    type uint64;
    config false;
    description
      "Number of packets received per second.";
  }
  leaf tx-bits-per-sec {
    type uint64;
    config false;
    description
      "Number of bits sent per second.";
  }
  leaf tx-pkt-per-sec {
    type uint64;
    config false;
    description
      "Number of packets sent per second.";
  }
  leaf rx-pkts {
    type uint64;
    config false;
    description
      "Total number of received packets.";
  }
  leaf rx-bytes {
    type uint64;
    config false;
    description
      "Total number of received bytes.";
  }
  leaf tx-pkts {
    type uint64;
    config false;
    description
      "Total number of sent packets.";
  }
  leaf tx-bytes {
```

```
        type uint64;
        config false;
        description
            "Total number of sent bytes.";
    }
    leaf rx-unicast-pkts {
        type uint64;
        config false;
        description
            "Number of received unicast packets.";
    }
    leaf rx-multicast-pkts {
        type uint64;
        config false;
        description
            "Number of received multicast packets.";
    }
    leaf rx-broadcast-pkts {
        type uint64;
        config false;
        description
            "Number of received broadcast packets.";
    }
    leaf drop-unicast-pkts {
        type uint64;
        config false;
        description
            "Number of discarded unicast packets.";
    }
    leaf drop-multicast-pkts {
        type uint64;
        config false;
        description
            "Number of discarded multicast packets.";
    }
    leaf drop-broadcast-pkts {
        type uint64;
        config false;
        description
            "Number of discarded broadcast packets.";
    }
    leaf tx-unicast-pkts {
        type uint64;
        config false;
        description
            "Number of sent unicast packets.";
    }
    leaf tx-multicast-pkts {
```

```

        type uint64;
        config false;
        description
            "Number of sent multicast packets.";
    }
    leaf tx-broadcast-pkts {
        type uint64;
        config false;
        description
            "Number of sent broadcast packets.";
    }
}

}

}

}

container vni-peer-infos {
    config false;
    description
        "The information table of vni members.";
    container peers {
        config false;
        description
            "The remote nve address in a same VNI.";
        list peer {
            key "vni-id source-ip peer-ip";
            config false;
            description
                "The remote nve address list in a same VNI.";
            leaf vni-id {
                type uint32 {
                    range "1..16777215";
                }
                config false;
                description
                    "The ID of VNI.";
            }
            leaf source-ip {
                type inet:ip-address-no-zone;
                config false;
                description
                    "The source address of the NVE interface.";
            }
            leaf peer-ip {
                type inet:ip-address-no-zone;
                config false;
                description

```



```
        "The remote NVE address.";
    }
    leaf tunnel-type {
        type peer-type;
        config false;
        description
            "Tunnel type.";
    }
    leaf out-vni-id {
        type uint32 {
            range "1..16777215";
        }
        config false;
        description
            "The ID of the out VNI.";
    }
}
}
}

container tunnel-infos {
    config false;
    description
        "VxLAN tunnel information.";
    list tunnel-info {
        key "tunnel-id";
        config false;
        description
            "VxLAN tunnel information list.";
        leaf tunnel-id {
            type uint32 {
                range "1..4294967295";
            }
            config false;
            description
                "The ID of Vxlan tunnel.";
        }
        leaf source-ip {
            type inet:ip-address-no-zone;
            config false;
            description
                "Local NVE interface address.";
        }
        leaf peer-ip {
            type inet:ip-address-no-zone;
            config false;
            description
                "Remote NVE interface address.";
```

```
    }
    leaf status {
        type tunnel-status;
        config false;
        description
            "Tunnel status.";
    }
    leaf type {
        type tunnel-type;
        config false;
        description
            "Tunnel type.";
    }
    leaf up-time {
        type string {
            length "1..10";
        }
        config false;
        description
            "Vxlan tunnel up time.";
    }
    leaf vrf-name {
        type leafref {
            path "/ni:network-instances/ni:network-instance/ni:name";
        }
        default "_public_";
        config false;
        description
            "The name of VPN instance.";
    }
}

augment "/if:interfaces/if:interface" {
    description
        "Augment the interface, NVE as an interface.";
    container nvo3-nve {
        when "if:interfaces/if:interface/if:type = 'Nve'";
        description
            "Network virtualization edge.";
        leaf source-vtep-ip {
            type inet:ipv4-address-no-zone;
            description
                "The source address of the NVE interface.";
        }
        leaf source-vtep-ipv6 {
            type inet:ipv6-address-no-zone;
            description
                "The source address of the NVE interface.";
```

```
        "The source ipv6 address of the NVE interface.";
    }
    leaf bypass-vtep-ip {
        type inet:ipv4-address-no-zone;
        description
            "The source address of bypass VXLAN tunnel.";
    }
    container statistics {
        description
            "VXLAN Tunnel Traffic Statistical Configuration Table.";
        list statistic {
            key "vni-id mode peer-ip direction";
            description
                "VXLAN Tunnel Traffic Statistics Configuration.";
            leaf vni-id {
                type uint32 {
                    range "1..16777215";
                }
                description
                    "ID of the VNI.";
            }
            leaf mode {
                type vni-type;
                description
                    "The type of the NVE interface.";
            }
            leaf peer-ip {
                type inet:ipv4-address-no-zone;
                description
                    "IP address of the remote VTEP.";
            }
            leaf direction {
                type direction-type;
                must "(./mode='l3' and ./bound!='bidirection')";
                description
                    "Traffic statistics type about the VXLAN tunnel.";
            }
        }
        container info {
            config false;
            description
                "Traffic statistics about the peer.";
            leaf rx-pkts {
                type uint64;
                config false;
                description
                    "Total number of received packets.";
            }
            leaf rx-bytes {
```

```
        type uint64;
        config false;
        description
            "Total number of received bytes.";
    }
    leaf tx-pkts {
        type uint64;
        config false;
        description
            "Total number of sent packets.";
    }
    leaf tx-bytes {
        type uint64;
        config false;
        description
            "Total number of sent bytes.";
    }
    leaf rx-unicast-pkts {
        type uint64;
        config false;
        description
            "Number of received unicast packets.";
    }
    leaf rx-multicast-pkts {
        type uint64;
        config false;
        description
            "Number of received multicast packets.";
    }
    leaf rx-broadcast-pkts {
        type uint64;
        config false;
        description
            "Number of received broadcast packets.";
    }
    leaf tx-unicast-pkts {
        type uint64;
        config false;
        description
            "Number of sent unicast packets.";
    }
    leaf tx-multicast-pkts {
        type uint64;
        config false;
        description
            "Number of sent multicast packets.";
    }
    leaf tx-broadcast-pkts {
```

```
        type uint64;
        config false;
        description
            "Number of sent broadcast packets.";
    }
    leaf drop-unicast-pkts {
        type uint64;
        config false;
        description
            "Number of discarded unicast packets.";
    }
    leaf drop-multicast-pkts {
        type uint64;
        config false;
        description
            "Number of discarded multicast packets.";
    }
    leaf drop-broadcast-pkts {
        type uint64;
        config false;
        description
            "Number of discarded broadcast packets.";
    }
    leaf rx-bits-per-sec {
        type uint64;
        config false;
        description
            "Number of bits received per second.";
    }
    leaf rx-pkt-per-sec {
        type uint64;
        config false;
        description
            "Number of packets received per second.";
    }
    leaf tx-bits-per-sec {
        type uint64;
        config false;
        description
            "Number of bits sent per second.";
    }
    leaf tx-pkt-per-sec {
        type uint64;
        config false;
        description
            "Number of packets sent per second.";
    }
}
```

```
    }
  }

}

container nvo3-gateway {
  when "if:interfaces/if:interface/if:type = 'Vbdif'";
  description
    "Enable anycast gateway.";
  leaf vxlan-anycast-gateway {
    type boolean;
    default "false";
    description
      "Enable vxlan anycast gateway.";
  }
}

}

augment "/ni:network-instances/ni:network-instance/ni:ni-type" +
  "/l3vpn:l3vpn/l3vpn:l3vpn" {
  description "Augment for l3vpn instance";
  container vni-lists {
    description "Vni list for l3vpn";
    list vni {
      key "vni-id";
      description
        "Vni for current l3vpn instance";
      leaf vni-id {
        type uint32 {
          range "1..16777215";
        }
        description
          "The id of VNI.";
      }
    }
  }
}

}

augment "/ni:network-instances/ni:network-instance/ni:ni-type" +
  "/l2vpn:l2vpn" {
  description "Augment for l2vpn instance";
  container vni-lists {
    description "Vni list for l2vpn";
    list vni {
      key "vni-id";
      description
        "Vni for current l2vpn instance";
      leaf vni-id {
        type uint32 {
```

```
        range "1..16777215";
    }
    description
        "The id of VNI.";
}
leaf split-horizon-mode {
    type vni-bind-type;
    default "hub-mode";
    description
        "Split horizon mode.";
}
leaf split-group {
    type string {
        length "1..31";
    }
    description
        "Split group name.";
}
}
}
}

rpc reset-vni-instance-statistic {
    description
        "Clear traffic statistics about the VNI.";
    input {
        leaf vni-id {
            type uint32 {
                range "1..16777215";
            }
            mandatory true;
            description
                "ID of the VNI.";
        }
    }
}

rpc reset-vni-peer-statistic {
    description
        "Clear traffic statistics about the VXLAN tunnel.";
    input {
        leaf vni-id {
            type uint32 {
                range "1..16777215";
            }
            mandatory true;
            description
                "ID of the VNI.";
        }
    }
}
```

```
    leaf mode {
        type vni-type;
        mandatory true;
        description
            "The type of vni memeber statistic.";
    }
    leaf peer-ip {
        type inet:ipv4-address-no-zone;
        mandatory true;
        description
            "IP address of the remote NVE interface.";
    }
    leaf direction{
        type direction-type;
        must "(./mode='mode-l3' and ./bound!='bidirection')";
        mandatory true;
        description
            "Traffic statistics type about the VXLAN tunnel.";
    }
}
}
```

<CODE ENDS>

#### 4. Security Considerations

This document raises no new security issues.

#### 5. IANA Considerations

The namespace URI defined in Section 3.3 need be registered in the IETF XML registry [RFC3688].

This document need to register the 'ietf-nvo3-base' YANG module in the YANG Module Names registry [RFC6020].

#### 6. Contributors

Haibo Wang  
Huawei  
Email: rainsword.wang@huawei.com

Yuan Gao  
Huawei  
Email: sean.gao@huawei.com

Gang Yan



Huawei  
Email: yangang@huawei.com

Mingui Zhang  
Huawei  
Email: zhangmingui@huawei.com

Yubao (Bob) Wang  
ZTE Corporation  
Email: yubao.wang2008@hotmail.com

Ruixue Wang  
China Mobile  
Email: wangruixue@chinamobile.com

Sijun Weng  
China Mobile  
Email: wengsijun@chinamobile.com

## 7. Acknowledgements

Authors would like to thank the comments and suggestions from Tao Han, Weilian Jiang.

## 8. References

### 8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC7364] T. Narten, E. Gray, et al, "Problem Statement: Overlays for Network Virtualization", draft-ietf-nvo3-overlay-problem-statement, working in progress.
- [RFC7365] Marc Lasserre, Florin Balus, et al, "Framework for DC Network Virtualization", draft-ietf-nvo3-framework, working in progress.
- [RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, August 2014.
- [I-D.ietf-nvo3-geneve] Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-

nvo3-geneve-10 (work in progress), March 2019.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC8014] D. Black, J. Hudson, L. Kreeger, M. Lasserre, T. Narten, An Architecture for Data-Center Network Virtualization over Layer 3 (NVO3), RFC8014, December 2016.

## 8.2. Informative References

[RFC7637] M. Sridharan, A. Greenberg, et al, "NVGRE: Network Virtualization using Generic Routing Encapsulation", RFC7637, September 2015.

[I-D.ietf-nvo3-vxlan-gpe] Maino, F., Kreeger, L., and U. Elzur, "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-06 (work in progress), April 2018.

[I-D.draft-ietf-bess-evpn-inter-subnet-forwarding] A. Sajassi, S. Salam, S. Thoria, J. Drake, J. Rabadan, "Integrated Routing and Bridging in EVPN", draft-ietf-bess-evpn-inter-subnet-forwarding-08, March 4, 2019.

[RFC8293] A. Ghanwani, L. Dunbar, V. Bannai, M. McBride, R. Krishnan, "A Framework for Multicast in Network Virtualization over Layer 3", RFC8293, January 2018.

Author's Addresses

Bing Liu  
Huawei Technologies  
No. 156 Beiqing Rd. Haidian District,  
Beijing 100095  
P.R. China

Email: [remy.liubing@huawei.com](mailto:remy.liubing@huawei.com)

Ran Chen  
ZTE Corporation

Email: [chen.ran@zte.com.cn](mailto:chen.ran@zte.com.cn)

Fengwei Qin  
China Mobile  
32 Xuanwumen West Ave, Xicheng District  
Beijing, Beijing 100053  
China

Email: [qinfengwei@chinamobile.com](mailto:qinfengwei@chinamobile.com)

Reshad Rahman  
Cisco Systems

Email: [rrahman@cisco.com](mailto:rrahman@cisco.com)