



Geneve TLV Parsing Example

Table of contents

Sample Code

```

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
+
|                                     ~ Variable-Length Option Data ~
|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
+

```

Sample Code

First we define the Geneve header, per [RFC 8926](#).

```

header geneve_t {
    bit<2> ver;
    bit<6> opt_len;
    bit<1> o;
    bit<1> c;
    bit<6> reserved1;
    bit<16> protocol_type;
    bit<24> vni;
    bit<8> reserved2;
};

```

Next we define a struct that contains the base Geneve Option header, shared by all option types.

```

// Struct so it can be a field within other headers
struct geneve_option_t {
    bit<24> option_class_type;
    bit<3> reserved;
    bit<5> length;
};

```

Lastly, depending on the option class and type, the variable length option data can be defined. In this example, we model the Geneve options used by the [P4 Inband Telemetry \(INT\) specification](#).

```
// Intermediate nodes (INT Transit Hops) must process this type of INT Header.
header geneve_option_int_md_t {
    geneve_option_t base;
    bit<4> version;
    bit<1> discard;
    bit<1> exceeded_max_hops;
    bit<1> mtu_exceeded;
    bit<12> reserved;
    bit<5> hop_ml;
    bit<8> remaining_hop_count;
    bit<16> instruction_bitmap;
    bit<16> ds_id;
    bit<16> ds_instruction;
    bit<16> ds_flags;
};

// Destination headers can be used to enable Edge-to-Edge communication between
// the INT Source and INT Sink.
header geneve_option_int_destination_t {
    geneve_option_t base;
    // Destination headers must only be consumed by the INT Sink
};

// Intermediate nodes (INT Transit Hops) must process this type of INT Header
// and generate reports to the monitoring system as instructed.
header geneve_option_int_mx_t {
    geneve_option_t base;
    bit<4> version;
    bit<1> discard;
    bit<27> reserved1;
    bit<16> instruction_bitmap;
    bit<16> ds_id;
    bit<16> ds_instruction;
```

```

    bit<16> ds_flags;
};

```

All the above headers and structs must be added to the application's headers struct, along with the NV_FIXED_HEADERS.

```

struct app_headers {
    NV_FIXED_HEADERS

    geneve_t custom_geneve;
    // geneve_option_t is not explicitly used by the P4 program, so a reference
    // placed in the headers the type is not optimized out. The NvOptionParser
    // instantiation references this type it by string name.
    geneve_option_t geneve_opt;
    geneve_option_int_md_t geneve_opt_int_md;
    geneve_option_int_destination_t geneve_opt_int_destination;
    geneve_option_int_mx_t geneve_opt_int_mx;
};

```

The last step for the parser is to define the parser state that will perform Geneve parsing. An extern object, NvOptionsParser, is used to further parse into the Geneve options. In this example, the TLV "type" is a combination of Option class and type:

- Class 0x103, type 1 (INT-MD)
- Class 0x103, type 2 (Destination-type)
- Class 0x103, type 3 (INT-MX)

```

parser geneve_parser(
    packet_in packet,
    out app_headers headers
) {
    NvOptionParser<bit<24>, _>(

```

```

    "opt_len",                // options_length_field
    2,                        // options_length_shift
    0,                        // options_length_add
    "geneve_option_t",       // option_layout_header_type
    "length",                 // option_length_field
    0,                        // option_length_shift
    4,                        // option_length_add
    "option_class_type",     // option_type_field
    (list<tuple<bit<24>, _>>){ // options
        {24w0x010301, "headers.geneve_opt_int_md"},
        {24w0x010302, "headers.geneve_opt_int_destination"},
        {24w0x010303, "headers.geneve_opt_int_mx"}
    }
) geneveOptions;

NV_FIXED_PARSER(packet, headers)

@nv_transition_from("nv_parse_udp", GENEVE_PORT)
state parse_geneve {
    // Fixed geneve is only an example; "base" header must be flex.
    packet.extract(headers.custom_geneve);
    geneveOptions.parseOptions(packet, headers);
    transition select(headers.custom_geneve.protocol_type) {
        NV_TYPE_IPV4 : nv_parse_inner_ipv4;
        NV_TYPE_IPV6 : nv_parse_inner_ipv6;
        NV_TYPE_MAC : nv_parse_inner_ethernet;
        default : accept;
    }
}
}
}

```

The control below shows how a match action table can be configured to now match on a specific INT domain, and perform per ID forwarding.

```

control int_over_geneve(
    inout app_headers headers,
    in nv_standard_metadata_t std_meta,
    inout nv_empty_metadata_t user_meta,
    inout nv_empty_metadata_t pkt_out_meta
) {
    action forward(bit<32> port) {
        nv_send_to_port(port);
    }

    action drop() {
        nv_drop();
    }

    table geneve_option_int_md_table {
        key = {
            headers.geneve_opt_int_md.ds_id : exact;
        }
        actions = {
            forward;
            NoAction;
        }
        default_action = NoAction();
        const entries = {
            (16w0x100) : forward(1);
        }
    }

    table geneve_option_int_destination_table {
        key = {
            headers.custom_geneve.vni : exact;
        }
        actions = {
            forward;
            NoAction;
        }
    }
}

```



```

    }
    default_action = NoAction();
    const entries = {
        (24w0x200) : forward(2);
    }
}

table geneve_option_int_mx_table {
    key = {
        headers.geneve_opt_int_mx.ds_id : exact;
    }
    actions = {
        forward;
        NoAction;
    }
    default_action = NoAction();
    const entries = {
        (16w0x300) : forward(3);
    }
}

apply {
    if (headers.custom_geneve.isValid()) {
        geneve_option_int_md_table.apply();
        geneve_option_int_destination_table.apply();
        geneve_option_int_mx_table.apply();
    }
    drop();
}
}

NvDocaPipeline(
    geneve_parser(),
    int_over_geneve()
) main;

```

See the full DPL example [geneve_tlv.p4](#)

Notice
This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation (“NVIDIA”) makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.
NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.
Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.
NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer (“Terms of Sale”). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.
NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer’s own risk.
NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer’s sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer’s product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.
No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.
Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.
THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, “MATERIALS”) ARE BEING PROVIDED “AS IS.” NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA’s aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.
Trademarks
NVIDIA and the NVIDIA logo are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

© Copyright 2025, NVIDIA. PDF Generated on 04/24/2025