Universal Serial Bus 4 (USB4™) Inter-Domain Service Protocol

Apple Inc.
Hewlett-Packard Inc.
Intel Corporation
Microsoft Corporation
Renesas Corporation
STMicroelectronics
Texas Instruments

Version 1.0

January 31, 2021

Version History

Revision	Comments	Issue Date
1.0	Initial release	January 2021

INTELLECTUAL PROPERTY DISCLAIMER

THIS SPECIFICATION IS PROVIDED TO YOU "AS IS" WITH NO WARRANTIES WHATSOEVER INCLUDING ANY WARRANTY OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE. THE AUTHORS OF THIS SPECIFICATION DISCLAIM ALL LIABILITY FOR INFRINGEMENT OF ANY PROPRIETARY RIGHTS, RELATING TO THE USE OR IMPLEMENTATION OF INFORMATION IN THIS SPECIFICATION. THE PROVISION OF THIS SPECIFICATION TO YOU DOES NOT PROVIDE YOU WITH ANY LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS.

Please send comments to techadmin@usb.org.

For industry information, refer to the USB Implementers Forum web page at http://www.usb.org.

Acknowledgement of Technical Contribution

The authors of this specification would like to recognize the following people who participated in the development of the USB4 Inter-Domain Service Protocol specification.

AMD			
Joe Scanlon			
Apple Inc.			
Clinton Bauder	Carlos Calderon	Thomas LaPerre	Rob McKeever
Collin Pieper	Reese Schreiber		
Intel Corporation			
Oren Bessudo	Mickey Gutman	Abdul Ismail	Brad Saunders
Uri Soloveychik	Stephanie Wallick	Gal Yedidia	
Microsoft Corporation			
Rahul Ramadas			

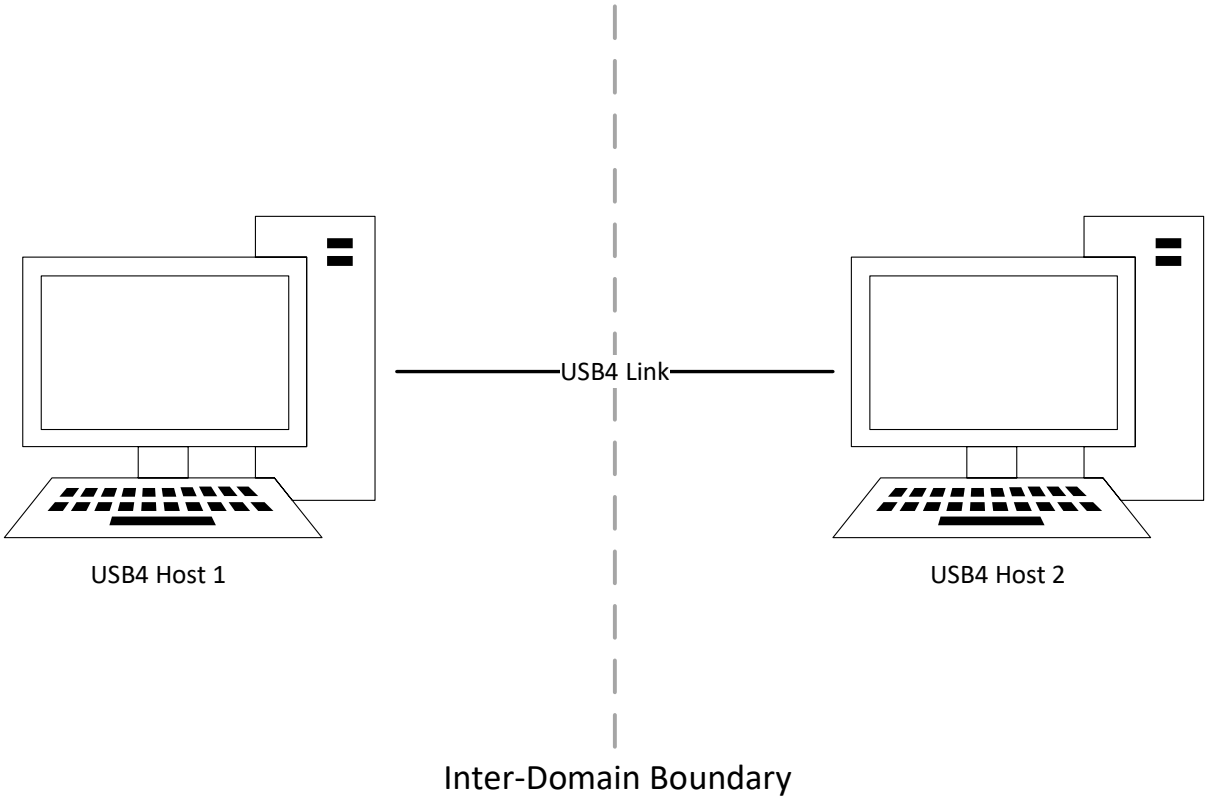
Figures

Tables

1 Introduction

USB4 is designed around a Domain that includes a Host, several Devices, and a Connection Manager running on the Host to manage the Domain. Although Domains are generally independent, USB4 allows for communication between Host Routers in adjacent Domains and between the Connection Managers in these Domains. The USB4 specification defines the Inter-Domain communication mechanism, but it does not specify how to set up a service to run over the Inter-Domain channel.

Figure 1-1. Connecting two USB4 Domains



USB4 provides the means to establish Inter-Domain Services between software stacks running in peer Domains. Tunneling of Ethernet traffic over USB4, defined in Chapter 5 of this specification, is an example of an Inter-Domain Service. Inter-Domain Services on peer Domains use the Host Interface in their local Host Routers to tunnel traffic through the USB4 fabric.

Inter-Domain Services are established using the Inter-Domain Protocol. The Inter-Domain Protocol is executed by peer Connection Managers, using Inter-Domain Control Packets for communication between the peer Connection Managers.

This specification defines the Inter-Domain Protocol and describes how to identify/setup an Inter-Domain Service using the Inter-Domain Protocol. It also defines the Inter-Domain Service to tunnel Ethernet traffic between USB4 Domains. Additional methods, outside the scope of this specification, may be needed to support Inter-Domain communication when more than two Domains are involved.

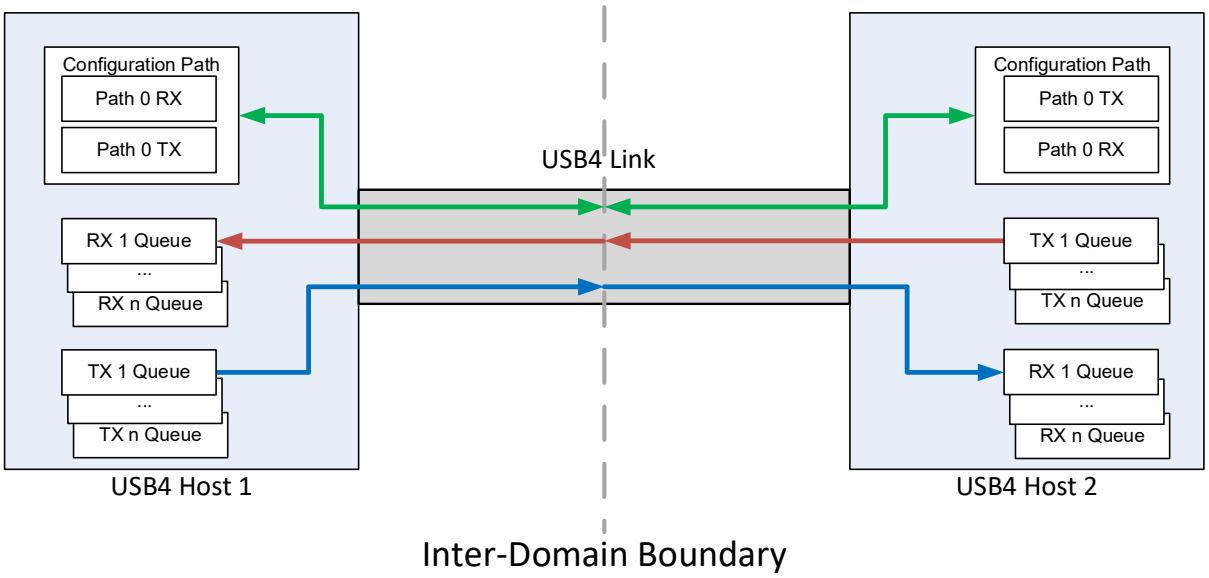
1.1 Inter-Domain Service Overview

During Domain enumeration, the Connection Manager discovers Inter-Domain Links using the flow described in the USB4 Connection Manager (CM) Guide.

Once the Local Connection Manager has discovered an Inter-Domain Link it uses the protocol defined in this specification to query the Remote Connection Manager of the other Domain about its capabilities. Responses are in the form of a series of Inter-Domain Packets, some of which contain a Properties Block as defined in Chapter 4. The Properties Block describes the Inter-Domain services a host offers. The Local Connection Manager then configures one or more

services published in the Properties Block. When service configuration is complete, one or more Tunneled Protocol Paths are established, and it is the responsibility of the overlaying protocol (e.g. Ethernet Protocol) to take over communication.

Figure 1-2. Inter-Domain Service Overview



1.2 Endian Conventions

All Control Packets sent across Path 0 meet ordering requirements as stated in the USB4 Specification. Endianness within the payload of a Tunneled Packet is defined by the protocol which is tunneled (see chapter 5).

UUID (universally unique identifier) values are written in canonical textual representation of 32 hexadecimal digits. For example, the UUID value of the Inter-Domain Discovery Protocol is B638D70E-42FF-40BB-97C2-90E2C0B2FF07. When written as part of a packet, the UUID values take four DWORDs, and the bytes within each DWORD are swapped. For example, the above UUID value is written as:

DWORD	Value
0	0x0ED738B6
1	0xBB40FF42
2	0xE290C297
3	0x07FFB2C0

1.3 Scope of the Specification

This specification is primarily targeted at platform operating system/BIOS/device driver developers that want to take advantage of the Inter-Domain communication capabilities of USB4.

1.4 Specification Organization

Chapter 1 provides an overview, Chapters 2, 3 and 4 contain technical information defining the Inter-Domain Discovery Protocol and Chapter 5 defines Inter-Domain Services (and specifically Ethernet over USB4).

1.5 Related Specifications

[USBCC] USB Type-C™ Cable and Connector Specification, Release 2.0 (USB Type-C Specification), August 2019

[USBPD] Universal Serial Bus Power Delivery Specification, Revision 3.0 Version 2.0 + ECNs, February 07, 2020 (USB PD Specification)

[USB4CM] Universal Serial Bus 4 (USB4™) Connection Manager Guide, Revision 1.0, date TBD

[USB4] Universal Serial Bus 4 (USB4™) Specification, Version 1.0, August 2019

1.6 Conventions

1.6.1 Precedence

If there is a conflict between text, figures, and tables, the precedence shall be tables, figures, and then text.

1.6.2 Keywords

The following keywords differentiate between the levels of requirements and options.

1.6.2.1 Informative

Informative is a keyword that describes information in this specification that intends to discuss and clarify requirements and features as opposed to mandating them.

1.6.2.2 May

May is a keyword that indicates a choice with no implied preference.

1.6.2.3 N/A

N/A is a keyword that indicates that a field or value is not applicable and has no defined value and shall not be checked or used by the recipient.

1.6.2.4 Normative

Normative is a keyword that describes features that are mandated by this specification.

1.6.2.5 Optional

Optional is a keyword that describes features not mandated by this specification. However, if an optional feature is implemented, the feature shall be implemented as defined by this specification (optional normative).

1.6.2.6 Reserved

Reserved is a keyword indicating reserved bits, bytes, words, fields, and code values that are set-aside for future standardization. The use and interpretation of these may be specified by future extensions to this specification and, unless otherwise stated, shall not be utilized or adapted by vendor implementation. A reserved bit, byte, word or field shall be set to zero by the sender and shall be ignored by the receiver. Reserved field values shall not be sent by the sender and, if received, shall be ignored by the receiver.

1.6.2.7 Shall

Shall is a keyword indicating a mandatory (normative) requirement. Designers are mandated to implement all such requirements to ensure interoperability with other compliant devices.

1.6.2.8 Should

Should is a keyword indicating flexibility of choice with a preferred alternative. Equivalent to the phrase "it is recommended that".

1.6.3 Capitalization

Some terms are capitalized to distinguish their definition in the context of this specification from their common English meaning. Words not capitalized have their common English meaning.

1.6.4 Italic Text

Italic text is used to identify variable names, register field and packet field names, or reference specification titles.

1.6.5 Numbering

Numbers that are immediately followed by a lowercase "b" (e.g., 01b) are binary values. Numbers that are immediately followed by an uppercase "B" are byte values. Numbers that are immediately followed by a lowercase "h" (e.g., 3Ah) are hexadecimal values. Numbers not immediately followed by either a "b", "B", or "h" are decimal values.

1.6.6 Bit, Byte, DW, and Symbol Conventions

A bit, byte, DW, or Symbol residing in location n within an array is denoted as bit(n), byte(n), DW(n), or Symbol(n).

A sequence of bits, bytes, DWs, or Symbols residing in locations n to m (inclusive) within an array is denoted as bit[m:n], byte[m:n], DW[m:n], or Symbol[m:n].

1.6.7 Implementation Notes

Implementation notes are not a normative part of this specification. They are included for clarification and illustration only. Implementation notes within this specification are enclosed in a box and set apart from the other text.

1.6.8 Pseudo Code

Throughout this specification, pseudo code is used to illustrate operating principles. Comments are demarcated by double forward slashes ("//"). The pseudo code conventions include:

```
if/else conditions;
   if (condition)
      // true operations
   else
      // false operations

for loops:
   for (conditions)
      // operations
```

1.7 Reserved Values and Fields

Unless otherwise specified, fields and values marked "Rsvd" shall be handled as described in Table 1-1.

Table 1-1. Rsvd Value and Field Handling

Type	Handling
Packet Values	A transmitter shall not use a value in this specification that is marked as "Rsvd". The target of a packet shall ignore a packet that has any of its defined fields set to a Rsvd value and proceed as if the packet was never received.

Type	Handling
Properties Block Fields	The Connection Manager shall not write a field with a value that is marked as “Rsvd”. Writing a field with a value that is marked as “Rsvd” results in undefined behavior.
Packet Fields	A transmitter shall set a field that is marked “Rsvd” to zero. A receiver shall ignore any fields that are marked “Rsvd”.
Host Interface Descriptor Fields	A Host shall only write a zero value to a field that is marked “Rsvd”. A Router may write any value in a field marked “Rsvd”.

1.8 Terms and Abbreviations

This section lists and defines the terms and abbreviations used throughout this specification. See the USB4 Specification for additional terms and abbreviations. Note that terms and abbreviations not defined here, use their generally accepted or dictionary meaning.

Term/Abbreviation	Description
Inter-Domain Packet	A Control Packet sent between peer Connection Managers.
Inter-Domain Discovery Protocol	The protocol used by peer Connection Managers to identify each other, probe their properties, and manage the Inter-Domain Link.
Inter-Domain Service	Sets up Paths between Connection Managers to allow tunneled Packets between Domains, carrying traffic for a certain service. Ethernet over USB4 (USB4NET) is an example of an Inter-Domain Service
Tunneled Protocol Path	A Path used for Tunneled protocol traffic over an Inter-Domain connection for an Inter-Domain Service. This Path uses a non-zero HopID.

2 Discovery Packet Formats

This section defines the structure of the Control Packets used by peer Connections Managers for Inter-Domain communication.

The target of an Inter-Domain Request packet shall ignore an Inter-Domain Request packet that has any of its defined fields set to a Rsvd value and proceed as if the packet was never received (see Table 1-1).

The target of an Inter-Domain Request packet shall ignore an Inter-Domain Request packet that has any of its defined fields set to an illegal value and proceed as if the packet was never received.

The target of an Inter-Domain Response packet shall ignore an Inter-Domain Response packet that has any of its defined fields set to a Rsvd value. The target shall invalidate the Inter-Domain Request packet that corresponds to the Inter-Domain Response packet. The target may reissue the Inter-Domain Request packet.

The target of an Inter-Domain Response packet shall ignore an Inter-Domain Response packet that has any of its defined fields set to an illegal value. The target shall invalidate the Inter-Domain Request packet that corresponds to the Inter-Domain Response packet. The target may reissue the Inter-Domain Request packet.

2.1 Inter-Domain Packet Format

Figure 2-1 defines the core structure of an Inter-Domain Control Packet. Subsequent sections describe the structure of a Control Packet based on the *PacketType* field.

Figure 2-1. Inter-Domain Packet Format

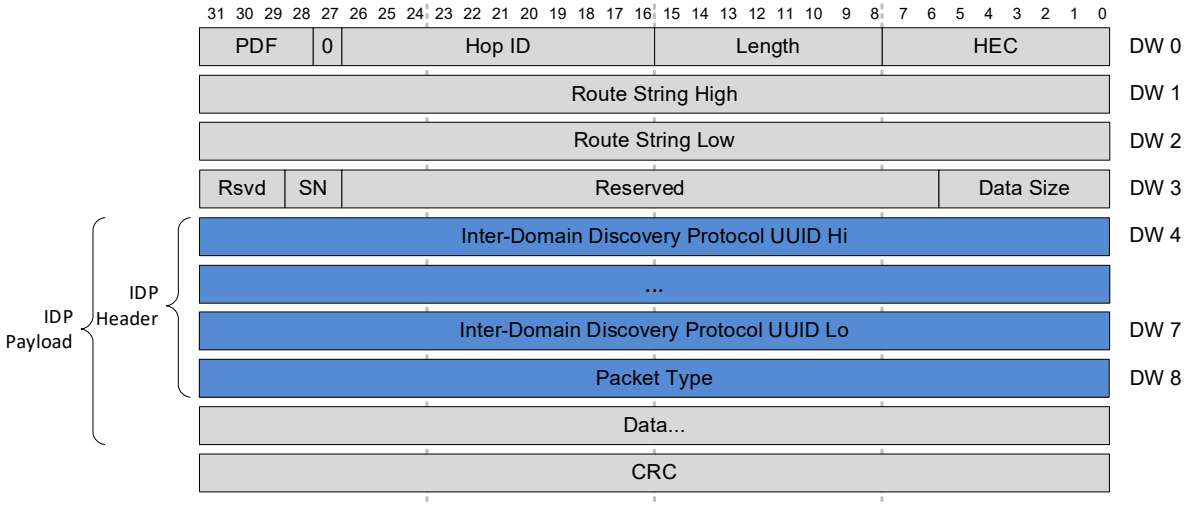


Table 2-1. Inter-Domain Packet Format Fields

DW	Bits	Description
0	31:0	**Header** – Transport Layer Packet header Bits: Description 7:0: HEC - Header Error Control 15:8: Length - Payload size in bytes excluding the padding size 26:16: HopID – Shall be set to 000h 27: SuppID – Shall be set to 0 31:28: PDF - Shall be set to 6 for an Inter-Domain Request or 7 for an Inter-Domain Response.
1	31:0	**Route String High**

<table>
<tr><th>DW</th><th>Bits</th><th>Description</th></tr>
<tr><td>2</td><td>31:0</td><td>Route String Low. See USB4.Table 6-1</td></tr>
<tr><td>3</td><td>5:0</td><td>Data Size –This field is used to indicate the number of doublewords in the Inter-Domain Packet Payload (IDPP) excluding the CRC.</td></tr>
<tr><td>3</td><td>28:27</td><td>SN – Sequence Number. A Connection Manager may set any value in an Inter-Domain Request Packet. The target of an Inter-Domain Request Packet shall write the value in the Request Packet into the Response Packet.</td></tr>
<tr><td>7:4</td><td>31:0</td><td>Inter-Domain Discovery Protocol UUID – A 128 bit UUID representing the Inter-Domain Discovery Protocol (B638D70E-42FF-40BB-97C2-90E2C0B2FF07)
<table>
<tr><th>DWORD</th><th>Value</th></tr>
<tr><td>4</td><td>0x0ED738B6</td></tr>
<tr><td>5</td><td>0xBB40FF42</td></tr>
<tr><td>6</td><td>0xE290C297</td></tr>
<tr><td>7</td><td>0x07FFB2C0</td></tr>
</table></td></tr>
<tr><td>8</td><td>31:0</td><td>Packet Type – The type of this packet as defined by Table 2-2</td></tr>
<tr><td>9:9+N -1</td><td>31:0</td><td>Data – The data payload (N DWords of data) based on the Packet Type</td></tr>
<tr><td>9+N</td><td>31:0</td><td>CRC – See Table 6-1 in the USB4 Specification for details on how this shall be calculated</td></tr>
</table>

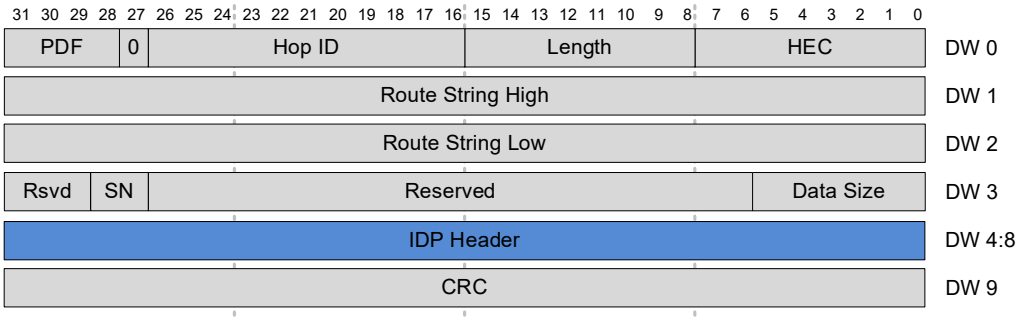
Table 2-2. Inter-Domain Packet Types

Value	Name	Description
0-1	Reserved	
2	UUID Response	A response to a UUID Request
3	Properties Read Request	A request for another Domain's Properties Block
4	Properties Read Response	A response to a Properties Read Request
5	Properties Changed Notification	Informs a peer Domain that the local Connection Manager has changed its Properties Block and may have modified the Inter-Domain services it offers
6	Properties Changed Response	A response to a Properties Changed Notification
7	Error Response	An error response to any packet type
8-11	Reserved	
12	UUID Request	A request for another Domain's UUID and route string
13-14	Reserved	
15	Link State Status Request	A request for another Domain's Target Link Width, Target Link Speeds, Supported Link Width, and Supported Link Speeds
16	Link State Status Response	A response to a Link State Status Request
17	Link State Change Request	A request for another Domain to modify its Target Link Width and Target Link Speeds
18	Link State Change Response	A response to a Link State Change Request
19+	Reserved	

2.2 Inter-Domain UUID Request Packet Format

Figure 2-2 defines the structure of an Inter-Domain UUID Request Packet.

Figure 2-2. Inter-Domain UUID Request Packet Format



2.3 Inter-Domain UUID Response Packet Format

Figure 2-3 defines the structure of an Inter-Domain UUID Response Packet.

Figure 2-3. Inter-Domain UUID Response Packet Format

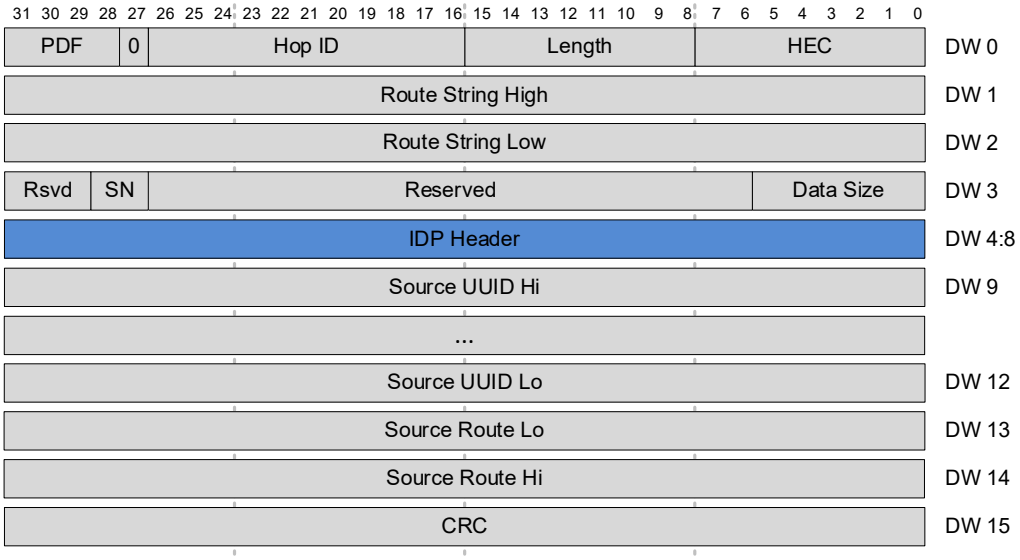


Table 2-3. Inter-Domain UUID Response Packet Fields

DW	Bits	Description
9:12	31:0	**Source UUID** – A 128 bit UUID representing the sender's Inter-Domain UUID, generated using the ISO/IEC 9834-8:2014 specification. Source UUID uniquely identifies a Domain.
13:14	31:0	**Source Route** – The sender's Route String. The *Source Route* field shall set to the value of the Route String field in the Inter-Domain UUID Request Packet In case of a loop inside a Domain, the Connection Manager may use this information to identify the location of the loop within the Domain.

2.4 Inter-Domain Properties Read Request Packet Format

Figure 2-4 defines the structure of an Inter-Domain Properties Read Request Packet.

Figure 2-4. Inter-Domain Properties Read Request Packet Format

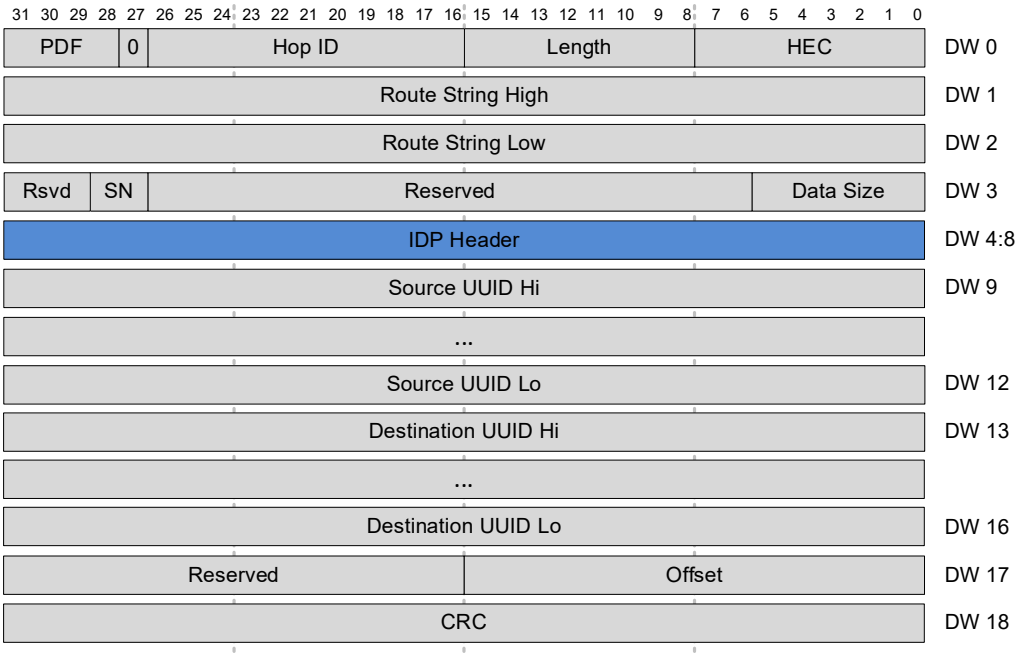


Table 2-4. Inter-Domain Properties Read Request Packet Fields

DW	Bits	Description
9:12	31:0	**Source UUID** – A 128 bit UUID representing the sender's Inter-Domain UUID
13:16	31:0	**Destination UUID** – A 128 bit UUID representing the receiver's Inter-Domain UUID.
17	15:0	**Offset** – The offset in DWORDS into the Properties Block that the Remote Connection Manager shall start populating in the appropriate field of the Inter-Domain Read Response Packet.
17	31:16	**Reserved**

2.5 Inter-Domain Properties Read Response Packet Format

Figure 2-5 defines the structure of an Inter-Domain Properties Read Response Packet.

Figure 2-5. Inter-Domain Properties Read Response Packet Format

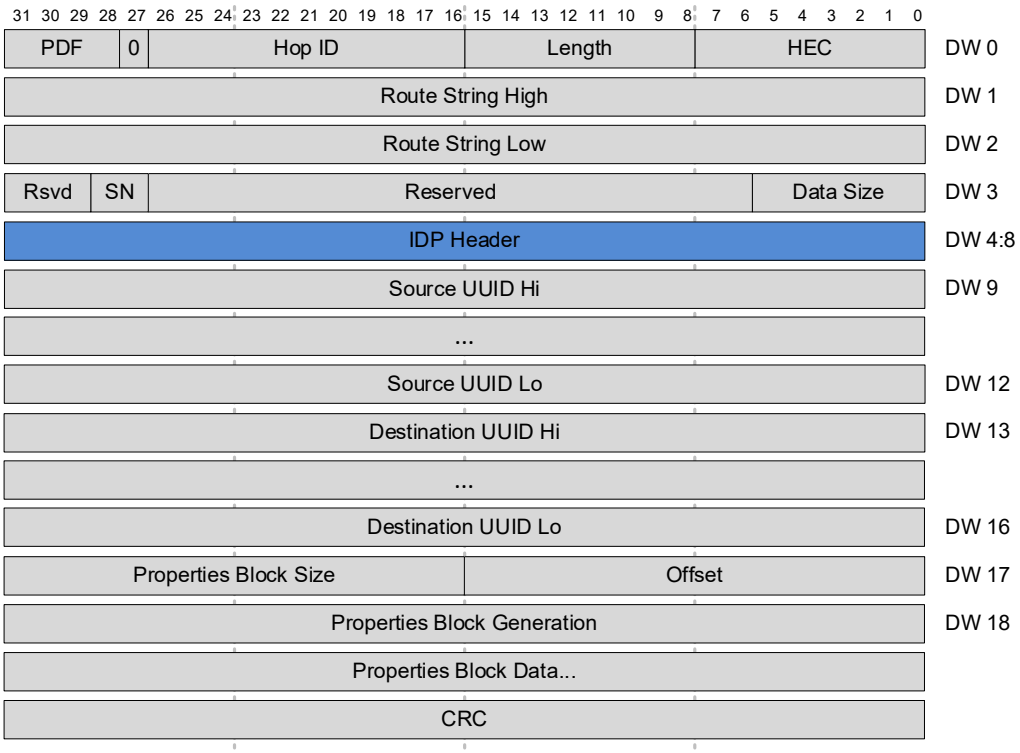


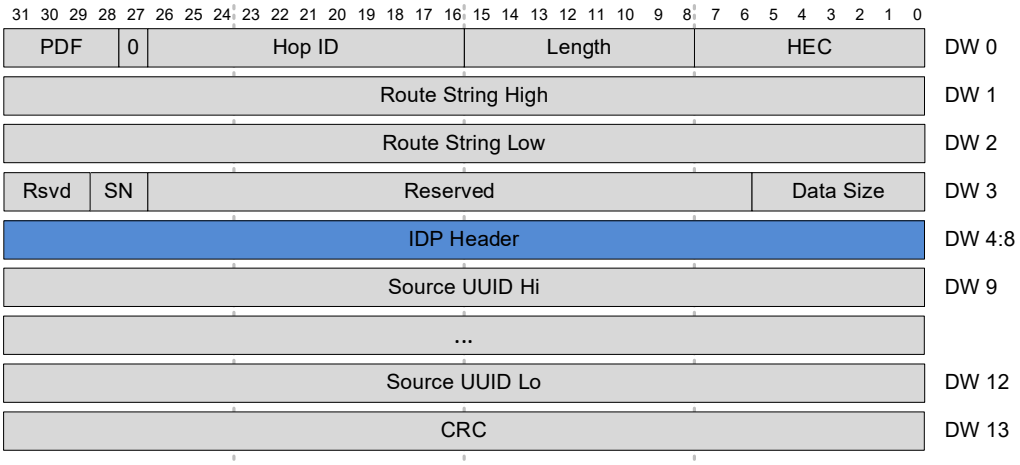
Table 2-5. Inter-Domain Properties Read Response Packet Fields

DW	Bits	Description
9:12	31:0	**Source UUID** – A 128 bit UUID representing the sender's Inter-Domain UUID
13:16	31:0	**Destination UUID** – A 128 bit UUID representing the receiver's Inter-Domain UUID. This field shall be set to the value received in the *Source UUID* field of the Inter-Domain Properties Read Request Packet.
17	15:0	**Offset** – The offset in DWORDS into the Properties Block that is delivered in this packet in the *Properties Block Data* field. This shall be equal to the *Offset* field in the Inter-Domain Properties Read Request Packet
17	31:16	**Properties Block Size** – The size, in DWORDS, of the entire Properties Block
18	31:0	**Properties Block Generation** – A monotonically incrementing number that tracks the generation of the Properties Block. Each time the local Properties Block is modified, the generation shall increment, and upon transmission, be reflected here. It is recommended that a new pseudo-random value is loaded each time the Connection Manager resets its Inter-Domain Discovery Protocol state. See Section 3.2 for usage of this field.
19:19 +(n-1)	31:0	**Properties Block Data** – The Properties Block data containing n DWs. Section 3.2 defines the rules for partitioning the Properties Block among multiple Inter-Domain Properties Read Response Packets.

2.6 Inter-Domain Properties Changed Notification Packet Format

Figure 2-6 defines the structure of an Inter-Domain Properties Changed Notification Packet.

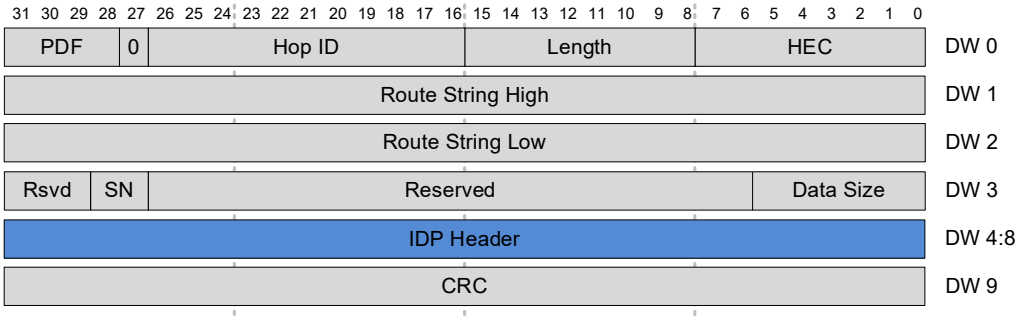
Figure 2-6. Inter-Domain Properties Changed Notification Packet Format



2.6.1 Inter-Domain Properties Changed Response Packet Format

Figure 2-7 defines the structure of an Inter-Domain Properties Changed Response Packet.

Figure 2-7. Inter-Domain Properties Changed Response Packet Format



2.7 Inter-Domain Error Response Packet Format

Figure 2-8 defines the structure of an Inter-Domain Error Response Packet.

Figure 2-8. Inter-Domain Error Response Packet Format

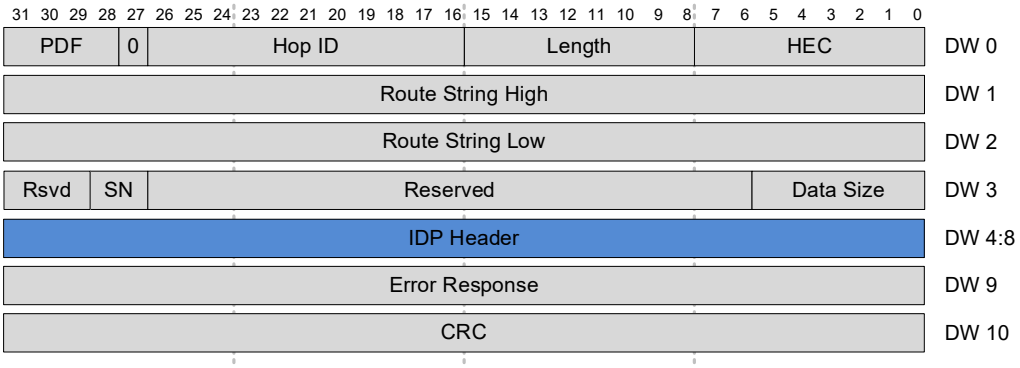


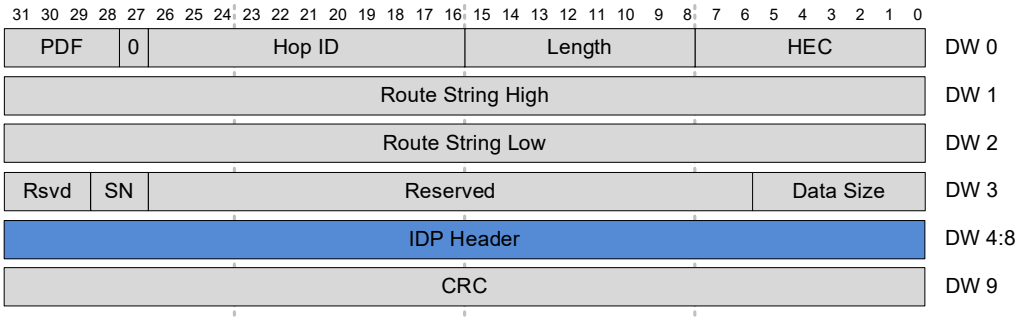
Table 2-6. Inter-Domain Error Response Packet Fields

DW	Bits	Description
9	31:0	**Error Response** Value: 1 – Unknown Packet Type; 4 – Not Ready; All other values – Reserved

2.8 Inter-Domain Link State Status Request Packet Format

Figure 2-9 defines the structure of an Inter-Domain Link State Status Request Packet.

Figure 2-9. Inter-Domain Link State Status Request Packet Format



2.9 Inter-Domain Link State Status Response Packet Format

Figure 2-10 defines the structure of an Inter-Domain Link State Status Response Packet.

Figure 2-10. Inter-Domain Link State Status Response Packet Format

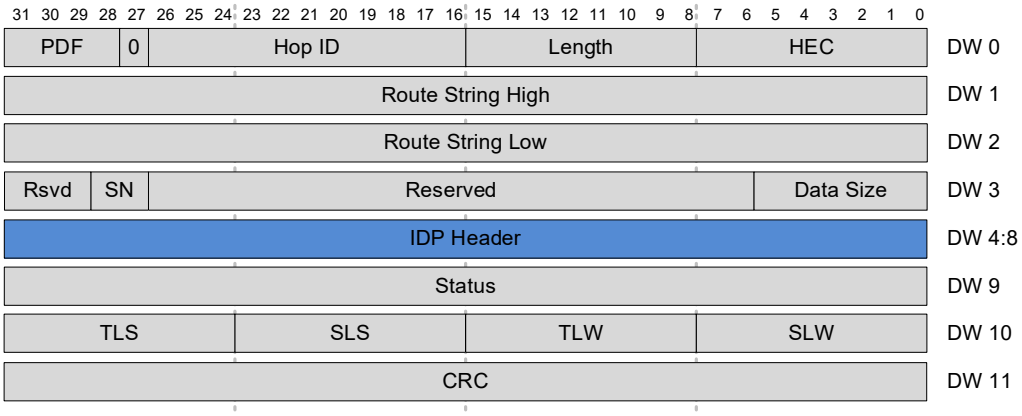


Table 2-7. Inter-Domain Link State Status Response Packet Fields

DW	Bits	Description
9	31:0	**Status** - shall be set to 0 to indicate successful response to the request
10	7:0	**Supported Link Width (SLW)** – Bits 25:20 of register LANE_ADP_CS_0[1]
10	15:8	**Target Link Width (TLW)** – Bits 9:4 of register LANE_ADP_CS_1
10	23:16	**Supported Link Speeds (SLS)** – Bits 19:16 of register LANE_ADP_CS _0

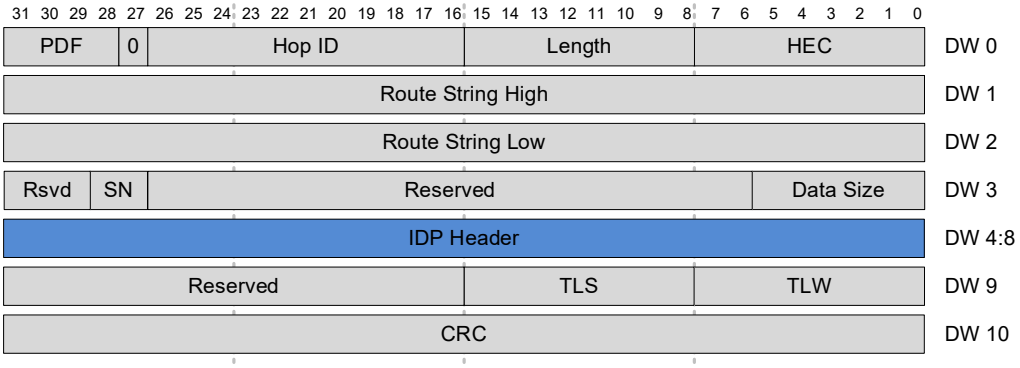
[1] The Lane Adapter Configuration Capability of the Adapter that is facing the Inter-Domain Link

DW	Bits	Description
10	31:24	**Target Link Speeds (TLS)** – Bits 3:0 of Register LANE_ADP_CS_1

2.10 Inter-Domain Link State Change Request Packet Format

Figure 2-11 defines the structure of an Inter-Domain Link State Change Request Packet.

Figure 2-11. Inter-Domain Link State Change Request Packet Format

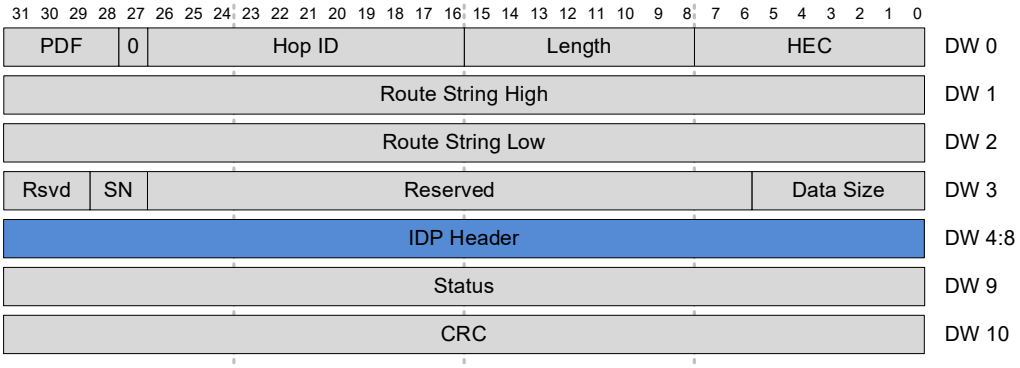


See Table 2-7 for definition of the *TLS* and *TLW* fields.

2.11 Inter-Domain Link State Change Response Packet Format

Figure 2-11 defines the structure of an Inter-Domain Link State Change Response Packet.

Figure 2-12. Inter-Domain Link State Change Response Packet Format



Refer to Table 2-7 for the definition of the *Status* field.

3 Service Discovery

3.1 Inter-Domain Discovery

The Local Connection Manager uses the mechanisms described in Section 3.2.1 and Section 3.3 of the USB4 Connection Manager Guide to identify an Inter-Domain Link.

Upon discovery of an Inter-Domain link, the Local Connection Manager issues an Inter-Domain UUID Request Packet to the Remote Connection Manager. If the Connection Manager does not receive an Inter-Domain UUID Response Packet within 1 second, it shall resend the request ten times, each time waiting for 1 second for a response. If no response is received, the Local Connection manager shall abort the Inter-Domain Discovery Protocol.

If Inter-Domain Discovery Protocol is aborted, and an Inter-Domain UUID Request Packet is later received, the recipient shall respond with an Inter-Domain UUID Response Packet. The recipient shall also issue an Inter-Domain UUID Request Packet to the peer in order to restart Inter-Domain Discovery.

During identification of an Inter-Domain Link each Connection Manager receives the 128-bit UUID value of the peer-Domain Connection Manager which can be used for unambiguous Domain identification.

3.2 Discovery of Peer Host Properties

Each Connection Manager sends an Inter-Domain Properties Read Request Packet and the receiver shall respond with an Inter-Domain Properties Read Response Packet. The Inter-Domain Properties Read Response Packet contains its Inter-Domain Properties Block (see Section 4.2 for the structure of the Properties Block).

If the Connection Manager does not receive an Inter-Domain Properties Read Response Packet within 1 second, it shall resend the request ten times, each time waiting for 1 second for a response. If no response is received, the Local Connection manager shall abort the Inter-Domain Discovery Protocol.

If the response does not contain the entire Properties Block (evident by comparing the *Data Size* field in the Inter-Domain Packet Header to the *Properties Block Size* field), the requester may ask for the next section of the Properties Block by issuing another Inter-Domain Properties Read Request Packet with the *offset* field advanced.

When the Properties Block is transferred using more than one Inter-Domain Properties Read Response Packet, all Inter-Domain Properties Read Response Packets but the last shall be maximum-length USB4 Packets.

When the Properties Block is transferred using more than one Inter-Domain Properties Read Response Packet, the receiving peer shall monitor the value of the *Properties Block Generation* field in successive Inter-Domain Properties Read Response Packets. If the value changes between successive Packets, the Connection manager shall read the Properties Block again from start.

3.3 Discover Change in Inter-Domain Capabilities

If a host's capabilities change (such as adding or removing new services) it shall inform any Inter-Domain peers by sending an Inter-Domain Properties Changed Notification Packet. Peers shall respond using an Inter-Domain Properties Changed Response Packet. If the Peer is interested in the changed capabilities, it shall send an Inter-Domain Properties Read Request Packet. The Connection Manager in the host that sent Inter-Domain Properties Changed Notification Packet responds with an Inter-Domain Properties Read Response as defined in Section 3.2. The Connection Manager shall increment the value in the *Properties Block Generation* field to indicate that the Properties Block has changed.

If the Connection Manager does not receive an Inter-Domain Properties Changed Response Packet within 1 second, it shall resend the request ten times, each time waiting for 1 second for a response.

A host that sends or receives an Inter-Domain Properties Changed Notification Packet shall terminate any existing services with the Domain not included in the updated Properties Block.

A host that sends or receives an Inter-Domain Properties Changed Notification Packet shall establish new services with the Domain included in the updated Properties Block.

3.4 Link Management of Lane Bonding

An Inter-Domain Link shall always operate in either Single Lane or Bonded mode. If both Inter-Domain peers support Lane Bonding, this configuration must be negotiated.

The Connection Manager of each host shall do the following:

- Exit Lane Bonding if the *Supported Link Width* field of the local Port facing the Inter-Domain Link is set to x1.
- Exit Lane Bonding if the Inter-Domain Link operated previously as a single-Lane Link, and the Connection Manager wishes to maintain the previous Link state.
- The Local Connection Manager shall issue a Link State Status Request Packet to its peer Domain. If the Connection Manager does not receive an Inter-Domain Link State Status Response Packet within 1 second, it shall resend the request ten times, each time waiting for 1 second for a response. If no response is received, the Local Connection manager shall exit Lane Bonding. Note that the peer Connection Manager may delay a response due to its current workload.
- Exit Lane Bonding if the *Supported Link Width* field received in the Link State Status Response Packet is set to x1.
- Compare the Local Domain's Inter-Domain UUID to the Remote Domain's Inter-Domain UUID. UUIDs are compared in canonical form, one byte at a time, starting from left to right. For example, a UUID of 44444444-AAAA-BBBB-444444444444 is larger than a UUID of 44334444-AAAA-BBBB-444444554444.
- The Local Domain that has a higher UUID shall:
    - Upon receiving a Link State Change Request Packet with the *Target Link Width* field set to 0x3, and if Lane Bonding is supported by the Port facing the Inter-Domain Link, set the *Target Link Width* field in both Lane Adapters of the Port to 0x3.
    - Upon receiving a Hot Plug Event Packet with the UPG bit set to 1b, read Lane 0 Adapter's *Negotiated Link Width* field (bits 25:20 of register LANE_ADP_CS_1) to verify that Lane Bonding is successful. Alternatively, poll the Lane 0 Adapter's *Negotiated Link Width* field for three seconds to wait for bonding to complete.
    - If bonding does not complete, assume single-Lane operation.
    - Exit Lane Bonding.
- The Local Domain that has a lower UUID shall:
    - Issue a Link State Change Request Packet with *Target Link Width* field set to 0x3 and the desired *Target Link Speed*. If the Connection Manager does not receive an Inter-Domain Link State Change Response Packet within 1 second, it shall resend the request ten times, each time waiting for 1 second for a response. If no response is received, the Local Connection manager shall abort Lane Bonding.
    - Issue a Link State Status Request. If a response is not received within 1 second or is received with a "Not Ready" status, resend the request ten times, each time waiting for 1 second for a response. If a response is received with an error, or if no response is received, the Local Connection manager shall abort Lane Bonding.
    - Exit Lane Bonding if the *Target Link Width* field in the Link State Status Response Packet does not have bit 1 set.
    - Set the Local *Target Link Width* field (bits 9:4 of register LANE_ADP_CS_1) to the maximum allowed value for both the Lane 0 and Lane 1 Adapters.

- Set the *Lane Bonding* bit (bit 15 in register LANE_ADP_CS_1) on either Lane Adapter to initiate Lane Bonding. Wait for the Lane 0 to reach CL0 state.

- Upon receiving a Hot Plug Event Packet with the UPG bit set to 1b, read Lane 0 Adapter's *Negotiated Link Width* field (bits 25:20 of register LANE_ADP_CS_1) to verify that Lane Bonding is successful. Alternatively, poll the Local Lane 0 Adapter's *Negotiated Link Width* field for *USB4.tBonding* to confirm the change.

If Lane Bonding fails, a Connection Manager shall disable Lane 1 as described in the Section 4.2.2 of the USB4 Specification.

An example flow is given in Appendix A.

3.5 Inter-Domain Error Recovery

If a Discovery Protocol packet is received with an unknown or reserved *Packet Type* field, the packet shall be discarded, and the receiver shall respond with an Error Response Packet containing an *Error Response* field set to "Unknown Packet Type".

If a Connection Manager receives a Link State Status Request Packet, it shall respond as soon as possible with the request values. However, if the necessary resources are unavailable, the Connection Manager may respond with the "Not Ready" status.

If a Connection Manager receives a Link State Change Request Packet, it shall service it as soon as possible by setting the requested values to each local Port of the Inter-Domain Link. However, if the necessary resources are unavailable, the Connection Manager may respond with the "Not Ready" status.

If an Error Response Packet is received, the receiver shall do the following depending on the *Error Response* field:

- *Not Ready*: Reissue the request up to ten times (one per second), and then fail the request if a proper response is not received.

- Unknown Packet Type: Fail the request

3.6 Termination of Discovery Protocol

When an Inter-Domain Link is disconnected, the Connection Manager shall terminate any Inter-Domain Services running with the disconnected Domain. It shall also initialize its Discovery Protocol state with the disconnected Domain.

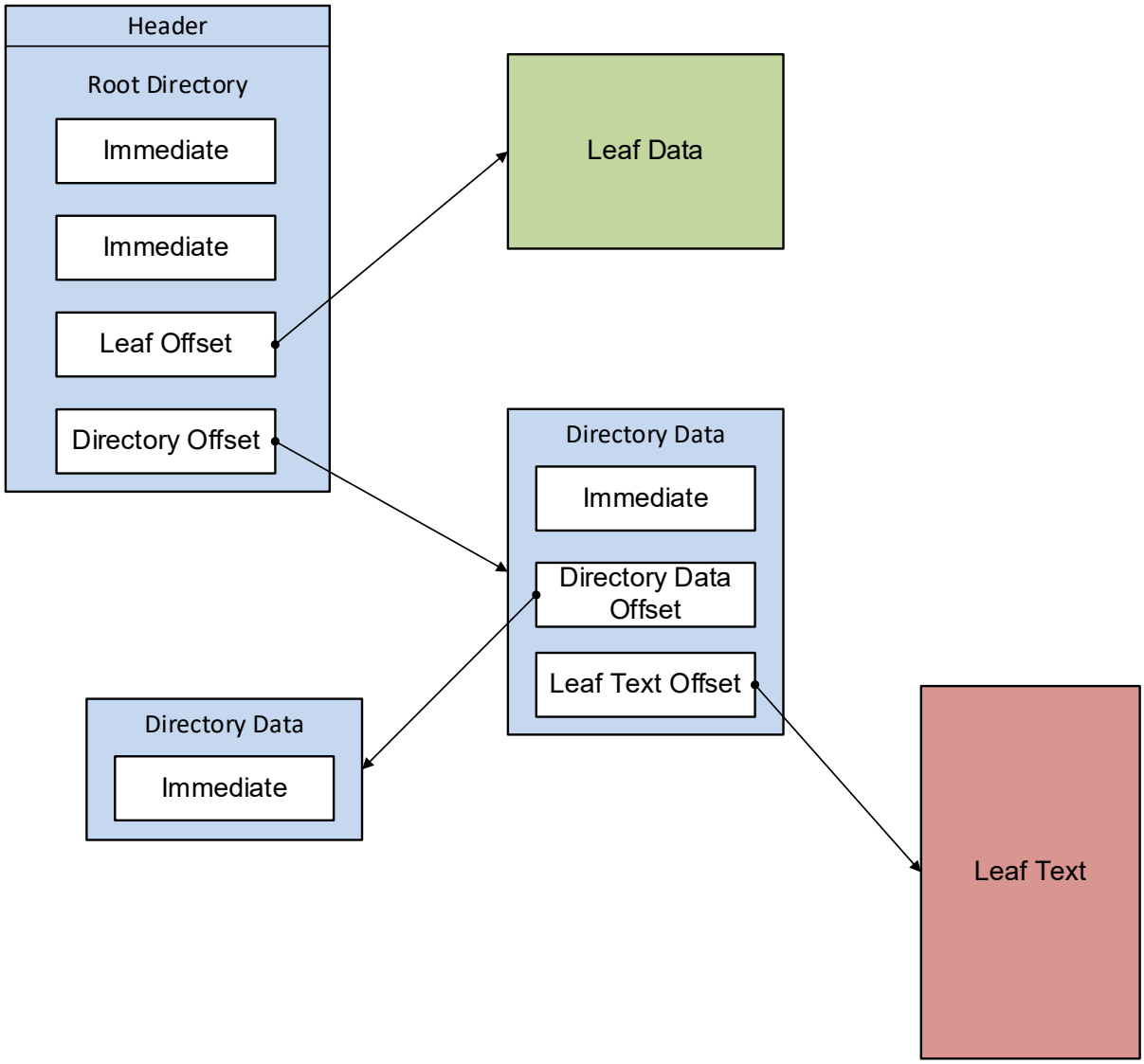
When a Connection Manager receives a Properties Changed Notification Packet on an Inter-Domain Link with a *Source UUID* field that is different from the *Source UUID* value established with the neighbor Domain on this Inter-Domain Link, the Connection Manager shall terminate any Inter-Domain Services running with the disconnected Domain. It shall also initialize its Discovery Protocol state with the disconnected Domain. The Connection Manager shall then start the Discovery Protocol with the neighbor Domain.

4 Inter-Domain Properties

This section defines the data structures and format of the Properties Block that describes a host's Inter-Domain properties. The Properties Block is transmitted in response to receiving an Inter-Domain Properties Read Request Packet.

This format is comprised of a Root Directory that provides basic identification information as well as optional Entries. These Entries may contain data or point to another Entry within the Properties Block.

Figure 4-1. Illustration of Typical Properties Block



4.1 Bit, byte, and DWORD ordering

Data structures defined in Section 4.2 appear in the order these are present within an Inter-Domain Properties Read Response Packet. DWORDs are listed in increasing order. Bits within each DWORD are presented in the order within the Packet; bit[0] within a DWORD of the Properties Block resides in bit[0] in the respective DWORD in the Inter-Domain Properties Read Response Packet.

4.2 Properties Block

The Properties Block shall be transmitted to a peer USB4 Host in response to a request for properties and shall include the size of the entire Properties Block, even if it has been spilt among various packet payloads.

Refer to Section 3.2 for more information on how to split the Properties Block.

4.2.1 Properties Block Header

Each Properties Block shall begin with two required DWORDs, “Version” and a “Required Fixed Value” first and “Root Directory Length” second.

Figure 4-2. Properties Block Header Format

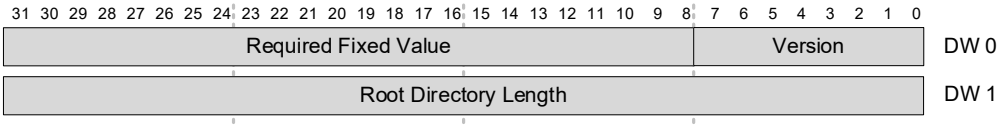


Table 4-1. Properties Block Header Fields

DW	Bits	Description
0	7:0	**Version** – This field shall be set to 0x01 for implementations conformant to this specification.
0	31:8	**Required Fixed Value**[2] – This field shall be set to 0x555844 for backwards compatibility reasons.
1	31:0	**Root Directory Length** – The length in DWORDs of the Root Directory contents. This length does not include DW0 and DW1.

The required fields are described in subsequent sections.

These fields, along with the required Entries described (See Section 4.2.4), are required within the Properties Block even if no optional properties Entries are added.

The hierarchical organization of the Properties Block is based upon a key-value paired Entry. Each key-value pair represents an elemental Entry such as a directory, a pointer (offset), or a value. This section specifies the format of these building blocks; their required usage is described later in this specification.

4.2.2 Generic Key-Value Entry format

Each Entry shall have the format shown in Figure 4-3.

Figure 4-3. Generic Key-Value Entry Format

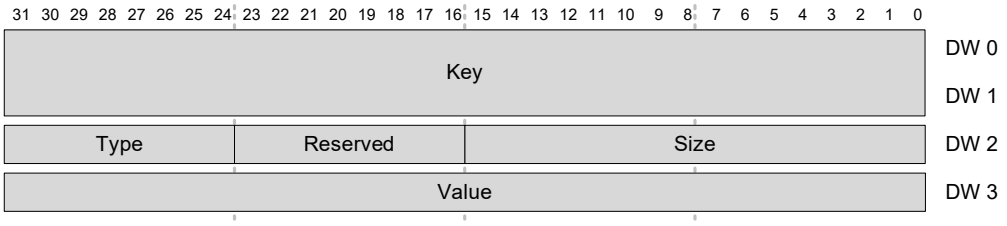


Table 4-2. Generic Key-Value Entry Fields

DW	Bits	Description
1:0	31:0	**Key** – A 64-bit value representing the Entry. The key shall relate to an 8-character ASCII code description of the Entry. For instance, 0x76656E646F726964 relates to ‘vendorid’. This value shall not be NULL terminated. 0x0 - 0xFFFFFFFFFFFF0000: Reserved for use by this specification. See Table 4-8 for a list of defined keys. 0xFFFFFFFFFFFF0001 - 0xFFFFFFFFFFFFFFFF: Vendor defined keys, see the Section 4.2.5.

[2] Historical Tidbit: This converts to ‘UXD’ which stood for “USB Cross Domain”

DW	Bits	Description
2	15:0	**Size** – The size of the value in regard to *Type* field - defined in more detail in the following sections.
2	23:16	**Reserved**
2	31:24	**Type** – A defined 8-bit value that specifies the version of the Entry. Defined types include:

Value	Type of Entry
0x00	Reserved
0x64 ('d')	Leaf Data Offset
0x74 ('t')	Leaf Text Offset
0x76 ('v')	Immediate Value
0x44 ('D')	Directory Data Offset
0x80 – 0xFF	Reserved for vendor-defined Types. See Section 4.2.5.1.

4.2.2.1 Immediate Value Entry format

All Immediate Value Entries shall have the format shown in Figure 4-4.

Figure 4-4. Immediate Value Entry Format

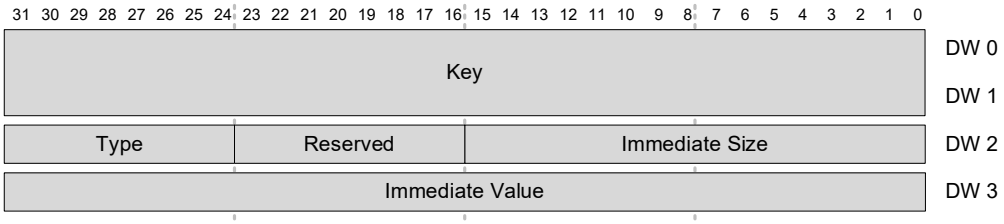


Table 4-3. Immediate Value Entry Fields

DW	Bits	Description
2	15:0	**Immediate Size** – Shall be set to 1.
2	31:24	**Type** – Immediate Value Entries shall use the type 0x76 ('v').
3	31:0	**Immediate Value** – A 32-bit value.

An Immediate Value Entry describes a scalar value of 32 bits or less. Interpretation of the value is defined per set of *Key* + *Type* fields.

4.2.2.2 Leaf Data Offset Entry format

All Leaf Data Offset Entries shall have the format shown in Figure 4-5.

Figure 4-5. Leaf Data Offset Entry Format

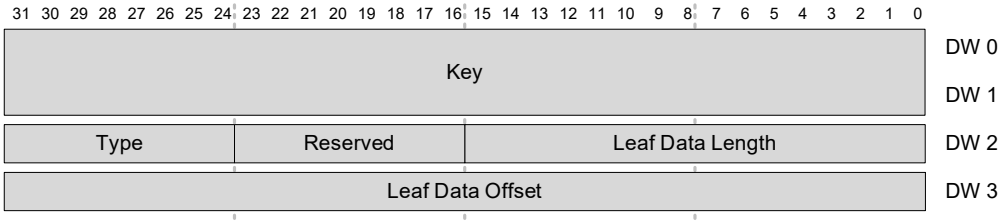


Table 4-4. Leaf Data Offset Entry Fields

DW	Bits	Description
2	15:0	**Leaf Data Length** – The number of DWORDs of leaf data which is pointed to by the leaf data offset.
2	31:24	**Type** – Leaf Data Offset Entries shall use the type 0x64 ('d').

DW	Bits	Description
3	31:0	**Leaf Data Offset** – The number of DWORDs from the start of the Properties Block (absolute offset) to a Leaf Data Entry that contains data of length specified by the *Leaf Data Length* field. The structure of the corresponding Leaf Data Entry is defined by the *Key* + *Type* fields.

4.2.2.3 Leaf Text Offset Entry format

All Leaf Text Offset Entries shall have the format shown in Figure 4-6.

Figure 4-6. Leaf Text Offset Entry Format

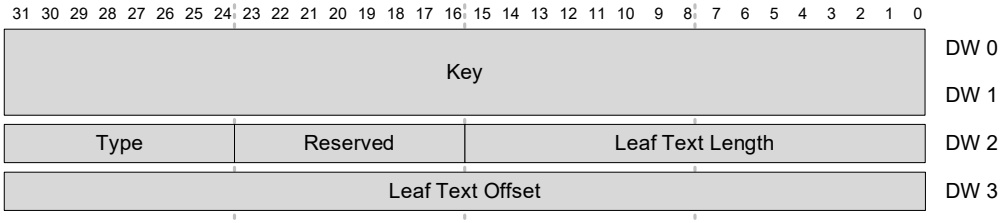


Table 4-5. Leaf Text Offset Entry Fields

DW	Bits	Description
2	15:0	**Leaf Text Length** – The number of DWORDs of leaf data which is pointed to by the leaf text offset.
2	31:24	**Type** – Leaf Text Offset Entries shall use the type 0x74 ('t').
3	31:0	**Leaf Text Offset** – The number of DWORDs from the start of the Properties Block (absolute offset) to the data that contains UTF-8 encoded text data of length specified by the *Leaf Text Length* field. Text data is required to be NULL terminated and zero padded to the next DWORD boundary. Interpretation of corresponding Leaf Text Entry is defined by the *Key* + *Type* fields.

4.2.2.4 Directory Data Offset Entry format

All Directory Data Offset Entries shall have the format shown in Figure 4-7.

Figure 4-7. Directory Data Offset Entry Format

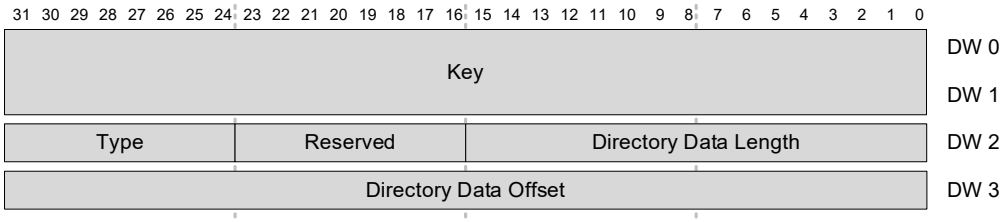


Table 4-6. Directory Data Offset Entry Fields

DW	Bits	Description
2	15:0	**Directory Data Length** – The number of DWORDs of directory content which is pointed to by the *Directory Data Offset* field. This length shall include the length of the directory UUID and Directory Data (see Figure 4-8), but not any data which is pointed to by an offset Entry within the directory referenced.
2	31:24	**Type** – Directory Data Offset Entries shall use the type 0x44 ('D').
3	31:0	**Directory Data Offset** – An offset from the start of the Properties Block that contains a directory, the size of its contents specified by the *Directory Data Length* field.

4.2.3 Directory Data Entry format

All Directory Data Entries shall have the format shown in Figure 4-8.

Figure 4-8. Directory Data Entry Format

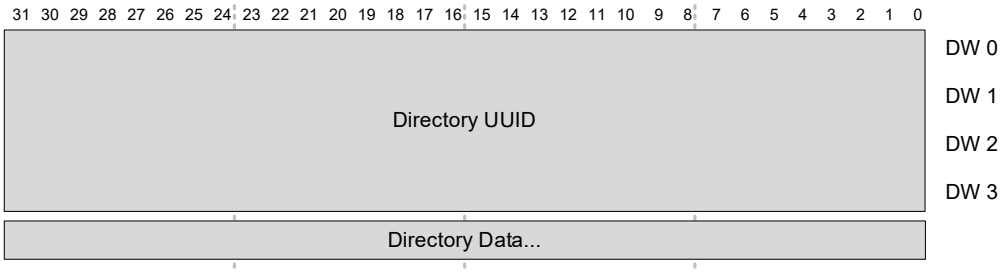


Table 4-7. Directory Data Entry Fields

DW	Bits	Description
3:0	31:0	**Directory UUID** – A 128 bit universally unique ID to identify the directory (Refer to ISO/IEC 9834-8:2014). This entry identifies a specific instance of a service. It can be used to tear down the service by removing the entry from the Properties Block and sending an Inter-Domain Properties Changed Notification Packet to the neighbor Connection Manager.
4	31:0	**Directory Data** – A directory that contains Entries and its size defined by its corresponding "Directory Data Length" minus the directory's UUID length.

Directory Data shall be contiguous (one Entry after another) and may begin at an offset after the preceding Entry.

A Directory Data Entry may include Immediate Value Entries, Leaf Data Offset Entries, Leaf Text Offset Entries, and Directory Data Offset Entries.

The Root Directory is defined as a standard Directory Data Entry except that it contains a Root Directory header (*Version* and *Root Directory Length* fields), instead of a *Directory UUID* header.

4.2.4 Required Entries

Each Properties Block is required to include the following Entries within the Root Directory:

- Vendor ID: An Immediate Value Entry with the key 'vendorid' which contains a 16-bit organizationally unique number that identifies the implementor of the Local Connection Manager. The USBIF is the registration authority and assigns these IDs.
- Device ID: An Immediate Value Entry with the key 'deviceid'. which identifies the Local Connection Manager software. This value is vendor defined.
- Max HopID: An Immediate Value Entry with the key 'maxhopid', which describes the maximum input HopID supported for an Inter-Domain connection

4.2.4.1 Miscellaneous

Offsets within Entries are not required to point to offsets in succession as they are ordered in the Properties Block.

4.2.4.2 Defined Keys

Table 4-8. Directory Key values

Key	ASCII Conversion	Description
0x76656E646F726964	vendorid	Vendor ID - An Entry that identifies the vendor of the Local Connection Manager software.
0x6465766963656964	deviceid	Device ID – A vendor specific Entry that further identifies the Local Connection Manager software.
0x6465766963657276	devicerv	Device Revision - A vendor-specific Entry that identifies the revision of the Local Connection Manager software.
0x6D6178686F706964	maxhopid	An Entry that describes the maximum input HopID supported for an Inter-Domain connection.

Key	ASCII Conversion	Description
0x7072746369640000	prtcid	An Entry that identifies the protocol described by the directory.
0x7072746376657273	prtcvers	An Entry that identifies the version of the protocol described by the directory.
0x7072746373746E73	prtcstns	An Entry that describes the settings of the software or hardware controlling the protocol described by the directory. May be useful as a bitmask.
0x7072746372657673	prtcrevs	A vendor-specific Entry that identifies the revision of the software controlling the protocol described by the directory.
0x6e6574776f726b00	network	A Directory Data Offset that points to a Directory Data Entry that contains information for Ethernet over USB4 protocol (USB4NET).
0x746172676574646D	targetdm	Reserved for future use.
0x6578746469737000	extdisp	Reserved for future use.
0x6175746873757000	authsup	Reserved for future use.

4.2.5 Vendor defined keys

0xFFFFFFFFFFFF0000 - 0xFFFFFFFFFFFFFFFF may be used as vendor defined keys within the Root Directory. These keys will be decoded based upon the *vendorid* and *deviceid* values present within the Root Directory.

0xFFFFFFFFFFFF0000 - 0xFFFFFFFFFFFFFFFF may be used as Directory-Data-Entry-specific keys when used within Directory Data Entries. These keys are decoded based upon the Directory Data Entry which contains these Entries.

Vendors are free to define the contents of these Vendor Defined Entries but must conform to the generic Entry format. Furthermore, if a defined type is used with a vendor-defined key, adherence to the Entry format for that defined type is required.

4.2.5.1 Vendor defined types

Vendors may define their own Entry types within the range 0x80 - 0xFF. These custom Entries must conform with the generic Entry format, see Table 4-2.

4.2.6 Protecting against buffer overruns

The Properties Block is required to be transmitted with the full size of the Property Block data buffer. This size should be taken into consideration when using the offset vectors so that decoding software does not overstep the bounds of the buffer that contains the Properties Block data.

Keys and types may have human readable conversions; however, they are not required to be NULL terminated. Software shall be careful not to overrun a buffer containing a character representation of a key or type using the length of each Entry and not look for NULL terminations.

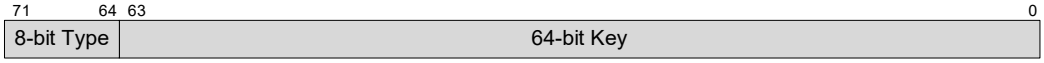
Leaf Text Entries are required to be NULL terminated, however, a string length is also provided so that software does not overrun this character buffer.

Leaf Data Entries and Directory Data Entries are provided with a size of the data which is referenced. This size is required to properly decode these structures.

4.2.7 Obtaining a one to one relationship between *Key* and Entry

The scope of a *Key* is within the Directory Entry that contains it. The same *Key* value may be used in different Directory Entries to identify different usages. Within one Directory Entry, a *Key* may be used multiple times, at most once with each *Type* field value. Therefore, when trying to establish a unique Entry ID within a Directory Entry, use a 72-bit ID by concatenating the *Type* and *Key* as shown in Figure 4-9.

Figure 4-9. Unique Directory Scope Entry ID



4.2.8 Discovery protocol layers

The Properties Block allows peer USB4 Domains to communicate to each other the basic service layers available above the USB4 native host interface layer. USB4 Inter-Domain Properties is designed to be a light-weight solution for the very simplest discovery of basic Inter-Domain protocols, but it does not prohibit more feature-rich solutions to be built on top of it.

4.2.9 Sample Properties Block

Table 4-9. Sample Properties Block

Offset	Value	Decoding
00	555844 01	Version = 1
01	00000018	Root Directory length (the Root Directory has 24 DWs content)
02	76656e64	"vend" (this is an Immediate entry containing the Vendor ID value)
03	6f726964	"orid"
04	76 00 0001	"v" R 1
05	00000a27	Immediate value, Vendor ID
06	76656e64	"vend" (this is a Leaf Text Offset entry pointing to a text string describing the vendor)
07	6f726964	"orid"
08	74 00 0003	"t" R 3
09	0000001a	Leaf Text Offset, (“Apple Inc.”)
0A	64657669	"devi" (this is an Immediate entry containing the Device ID value)
0B	63656964	"ceid"
0C	76 00 0001	"v" R 1
0D	0000000a	Immediate value, Device ID
0E	64657669	"devi" (this is a Leaf Text Offset entry pointing to a text string describing the device)
0F	63656964	"ceid"
10	74 00 0003	"t" R 3
11	0000001d	Leaf Text Offset, (“Macintosh”)
12	64657669	"devi" (this is an Immediate entry containing the Revision ID value)
13	63657276	"cerv"
14	76 00 0001	"v" R 1
15	80000100	Immediate value, Device Revision
16	6e657477	"netw" (a Directory Data Offset to Directory Data of type “network”)
17	6f726b00	"ork"
18	44 00 0014	"D" R 20

Offset	Value	Decoding
19	00000021	Directory data offset, (Network Directory)
		(End of Root Directory)
1A	4170706c	"Appl" (A Leaf Text entry)
1B	6520496e	"e In"
1C	632e0000	"c."
1D	4d616369	"Maci" (A Leaf Text entry)
1E	6e746f73	"ntos"
1F	68000000	"h"
20	00000000	Padding[3]
		Start of a Directory Data entry of type "network"
21	ca8961c6	Directory UUID, Network Directory
22	9541ce1c	Directory UUID, Network Directory
23	5949b8bd	Directory UUID, Network Directory
24	4f5a5f2e	Directory UUID, Network Directory
25	70727463	"prtc" (Immediate entry that identifies the protocol described by the directory)
26	69640000	"id"
27	76 00 0001	"v" R 1
28	00000001	Immediate value, Network Protocol ID
29	70727463	"prtc"
2A	76657273	"vers"
2B	76 00 0001	"v" R 1
2C	00000001	Immediate value, Network Protocol Version
2D	70727463	"prtc"
2E	72657673	"revs"
2F	76 00 0001	"v" R 1
30	00000001	Immediate value, USB4NET device driver Revision
31	70727463	"prtc"
32	73746e73	"stns"
33	76 00 0001	"v" R 1
34	00000000	Immediate value, Network Protocol Settings

[3] This is not required, but illustrates that it is possible to do so

5 Service Configuration

Once Service Discovery is completed, each Connection Manager is aware of its peer Domain's available Inter-Domain Services. To use a service, a protocol UUID and a protocol specific Login Packet is defined, which is used to initiate the configuration and establishment of USB4 Paths. Service drivers use their protocol UUID, encoded into each Inter-Domain packet, as a mechanism for determining which Inter-Domain packets are addressed to each service that has been configured. If a Connection Manager receives an Inter-Domain packet with a protocol UUID that it does not support, it shall ignore the packet.

A single instance of each supported Inter-Domain Service may be set between peer Connection Managers.

5.1 Ethernet over USB4 (USB4NET)

This section defines how to tunnel Ethernet traffic over USB4 (USB4NET).

A USB4NET Inter-Domain Service may be established after both Connection managers include a proper 'network' Directory Data Entry in their Property Blocks. Each Connection Manager sends an Inter-Domain USB4NET Login Packet to the peer Domain. Upon receipt of an Inter-Domain USB4NET Login Packet, the peer Connection Manager shall send an Inter-Domain USB4NET Login Response Packet or an Error Response Packet. If the Connection Manager does not receive an Inter-Domain Login Response Packet, it shall resend the request for at least 1 minute or till a response is received. It is recommended that Inter-Domain USB4NET Login Packets are first sent frequently (e.g. every 100ms) and later at a reduced rate (e.g. every 1 second). If no response is received, the Local Connection manager shall abort the USB4NET login.

If a USB4NET login is aborted, and an Inter-Domain USB4NET Login Packet is later received, the recipient shall respond with an Inter-Domain USB4NET Login Response Packet. The recipient shall also issue an Inter-Domain USB4NET Login Packet to the peer in order to establish a USB4NET Service.

Once a successful Inter-Domain USB4NET Login Response Packet is transmitted, the Connection Manager may set and enable a Path from the Inter-Domain Link that is associated with the Packet to a Receive Descriptor Ring in the Host Router's Host Interface.

Upon reception of an Inter-Domain USB4NET Login Response Packet, a Connection Manager may set an outbound Path from a Transmit Descriptor Ring in the Host Interface Adapter of its Host Router to the Lane Adapter that has been identified as the Inter-Domain boundary.

The Connection Manager may delay setting the inbound and outbound Paths and enabling the respective Descriptor Rings till a successful Inter-Domain USB4NET Login Response Packet is transmitted and an Inter-Domain USB4NET Login Response Packet is received, At that time, the Connection Manager shall set both Paths and enable the Descriptor Rings, if it has not done so already.

A Connection Manager may terminate a USB4NET Service by sending an Inter-Domain USB4NET Logout Packet to the peer Domain. This allows the peer Connection Manager the opportunity to cleanly terminate its service(s). The peer Connection Manager shall then respond with an Inter-Domain USB4NET Logout Response Packet. If the Connection Manager does not receive an Inter-Domain Logout Response Packet within 1 second, it shall resend the request ten times, each time waiting for 1 second for a response. If no response is received, it is assumed to have been received and shutdown proceeds.

Once a successful Inter-Domain USB4NET Logout Response Packet is transmitted, the responding Connection Manager shall tear down the Path and shall disable the Receive Descriptor Ring associated with the defunct login.

Upon reception of an Inter-Domain USB4NET Logout Response Packet, a Connection Manager shall tear down the Path and shall disable the Transmit Descriptor Ring associated with the defunct login.

Should the USB4 Link, Paths, or USB4NET Service be terminated with traffic still pending transmission, all traffic not posted yet to a Transmit Descriptor Ring is failed back to the network stack, if appropriate, or dropped.

5.1.1 Directory Data Entry

The Properties Block includes a USB4NET Directory Data Entry for USB4NET. A Directory Data Offset Entry in the Root Directory Entry points to the USB4NET Directory Data Entry.

A USB4NET Directory Data Entry contains the following:

- A Directory UUID with contents that are unique to this Directory Data Entry.
- An Immediate Value Entry of *Type* = prtcid with value = 0x1 (protocol is USB4NET)
- An Immediate Value Entry of *Type* = prtcvers

Table 5-1. Version of USB4NET Protocol (prtcvers)

Value	Decoding
0x1	Version included in Rev. 1.0 of this specification
All other values	Reserved

- An Immediate Value Entry of *Type* = prtcstns. The Immediate Value field is a bit vector with **contents described in Table 5-2**

Table 5-2. Decoding of prtcstns Immediate Value field

Bit	Description
0	End-to-End (E2E) Flow Control is supported for USB4NET traffic. 0b – Not supported 1b - Supported It is recommended to support End-to-End Flow Control.
1	This host supports the USB4NET Frame format, including the *Frame ID* field 0b – Not supported 1b – Supported Shall be set to 1b.
2	This host supports segmentation of Ethernet frames with payload of up to 64KB into USB4 Frames. 0b –Host supports reception of up to 64KB Ethernet frames. It is recommended that host only sends Ethernet frames of up to 1500 bytes, but it may deliver Ethernet frames of up to 64KB. 1b – Host supports 64KB Ethernet frames on transmit and receive. See Section 5.1.3 for more details.
3-31	Reserved

5.1.2 Packet Types

5.1.2.1 Inter-Domain USB4NET Packet

Figure 5-1 defines the structure of a generic Inter-Domain USB4NET Packet.

Figure 5-1. Inter-Domain USB4NET Packet Format

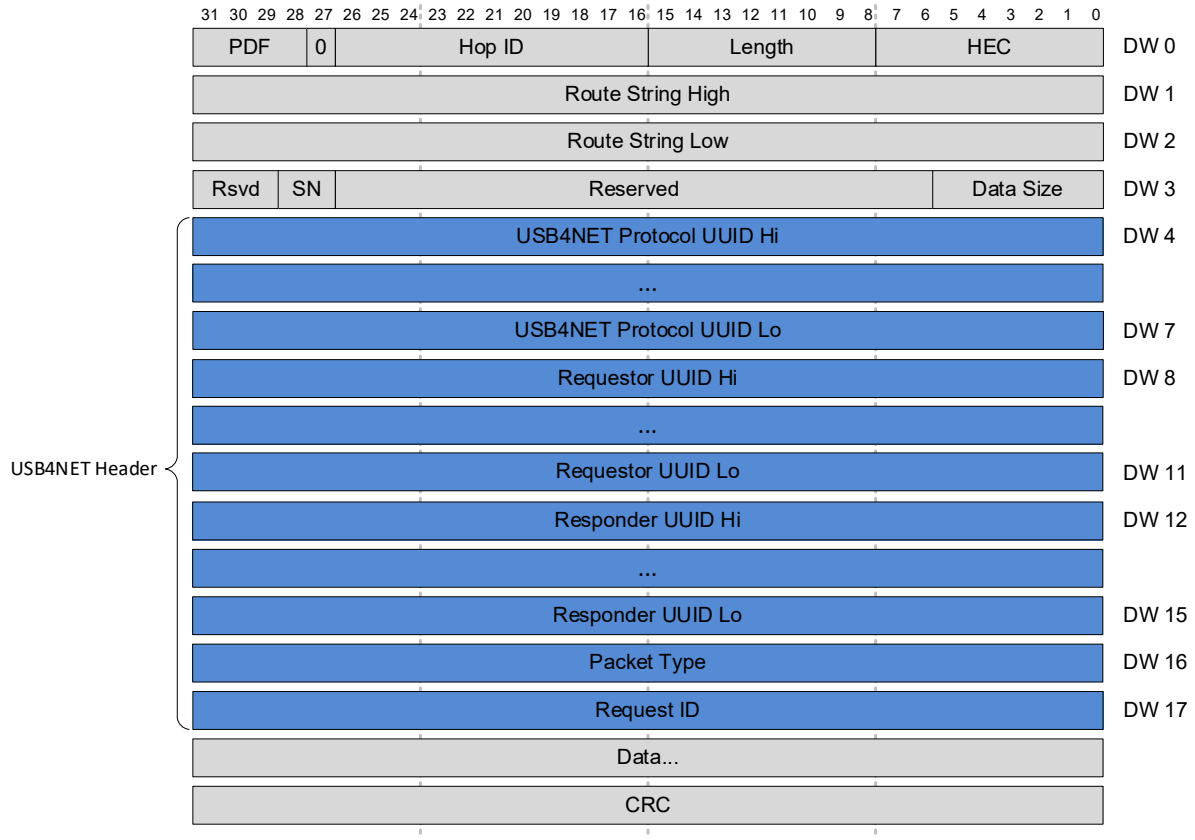


Table 5-3. Inter-Domain USB4NET Packet Fields

<table>
<tr><th>DW</th><th>Bits</th><th>Description</th></tr>
<tr><td>0</td><td>31:0</td><td>Header – Transport Layer Packet header
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>7:0</td><td>HEC - Header Error Control</td></tr>
<tr><td>15:8</td><td>Length - Payload size in bytes excluding the padding size</td></tr>
<tr><td>26:16</td><td>HopID – Shall be set to 000h</td></tr>
<tr><td>27</td><td>SuppID – Shall be set to 0</td></tr>
<tr><td>31:28</td><td>PDF - Shall be set to 6 or 7</td></tr>
</table></td></tr>
<tr><td>1</td><td>31:0</td><td>Route String High</td></tr>
<tr><td>2</td><td>31:0</td><td>Route String Low. See USB4.Table 6-1</td></tr>
<tr><td>3</td><td>5:0</td><td>Data Size –This field is used to indicate the number of doublewords in the Inter-Domain Packet Payload (IDPP) excluding the CRC.</td></tr>
<tr><td>3</td><td>28:27</td><td>SN – Sequence Number. Not used, a Connection Manager may assign any value.</td></tr>
<tr><td>4:7</td><td>31:0</td><td>USB4NET Protocol UUID – A 128 Bit UUID representing the USB4NET protocol, 798F589E-3616-8A47-97C6-5664A920C8DD
<table>
<tr><th>DWORD</th><th>Value</th></tr>
<tr><td>4</td><td>0x9E588F79</td></tr>
<tr><td>5</td><td>0x478A1636</td></tr>
<tr><td>6</td><td>0x6456C697</td></tr>
<tr><td>7</td><td>0xDDC820A9</td></tr>
</table></td></tr>
<tr><td>8:11</td><td>31:0</td><td>Requestor UUID – A 128 Bit UUID representing the Requestor’s Inter-Domain UUID</td></tr>
<tr><td>12:15</td><td>31:0</td><td>Responder UUID – A 128 Bit UUID representing the Responder’s Inter-Domain UUID</td></tr>
</table>

DW	Bits	Description
16	31:0	**Packet Type**– A USB4NET packet type, as defined by Table 5-4
17	31:0	**Request ID**– An incrementing, sequential 32-bit value used to match packet responses. Shall be incremented per each Request Packet sent. Shall be copied from each Request Packet to the same field in the Response Packet.

Table 5-4. Inter-Domain USB4NET Packet Types

Value	Name	Description
0	Login	Sent from the Requestor to the Responder. This packet requests a login of a USB4NET Service with the Responder.
1	Login Response	A response from the Responder to the Requestor of a Login.
2	Logout	Sent from the Requestor to cancel a USB4NET Service with the Responder.
3	Logout Response	A response from the Responder to the Requestor of a Logout.
All other values	Reserved	Reserved

5.1.2.2 Inter-Domain USB4NET Login Request Packet

Figure 5-2 defines the structure of an Inter-Domain USB4NET Login Request Packet.

Figure 5-2. Inter-Domain USB4NET Login Request Packet Format

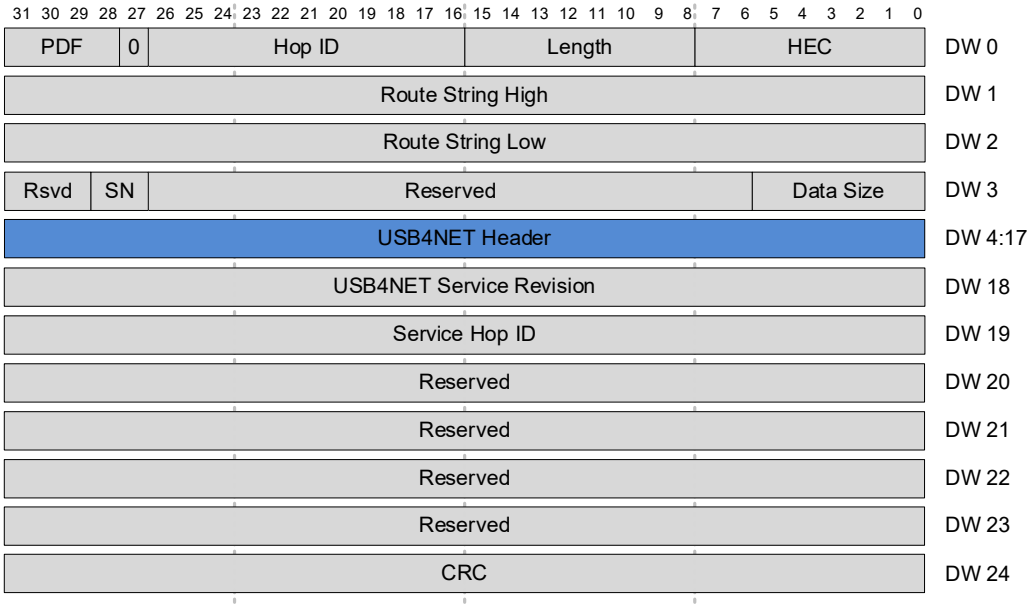


Table 5-5. Inter-Domain USB4NET Login Request Packet Fields

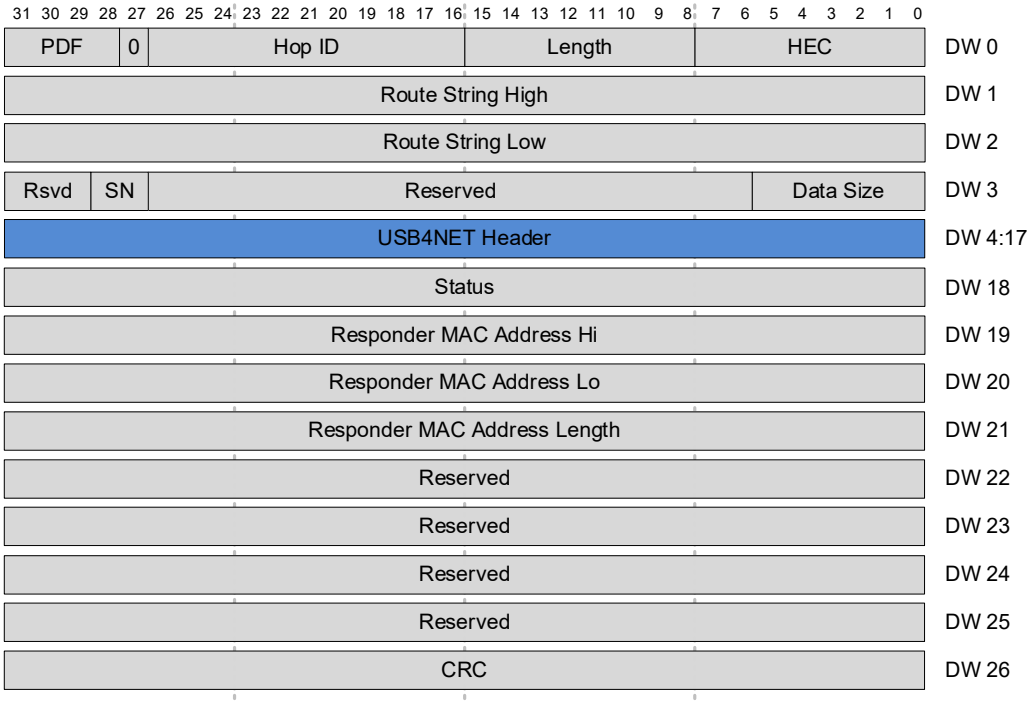
DW	Bits	Description
18	31:0	**USB4NET Service Revision** – A value representing the revision of the USB4NET Service protocol. Set to 1.

DW	Bits	Description
19	31:0	**Service Hop ID** – This is the HopID used by this service when transmitting packets from this domain to its peer domain. The Service Hop ID shall not exceed the value reported by the peer Domain in its Properties Block. To support a TBT3-compatible neighbor Connection Manager, if the Properties Block read from the neighbor Connection Manager does not contain a maxhopid Entry, then the Router shall use HopID=15 for this service when transmitting packets from this domain to its peer domain.

5.1.2.3 Inter-Domain USB4NET Login Response Packet

Figure 5-3 defines the structure of an Inter-Domain USB4NET Login Response Packet.

Figure 5-3. Inter-Domain USB4NET Login Response Packet Format



If the Responder already has a login with another Requestor or otherwise cannot process this request it shall return a failure indication in the Status field of the Inter-Domain USB4NET Login Response Packet.

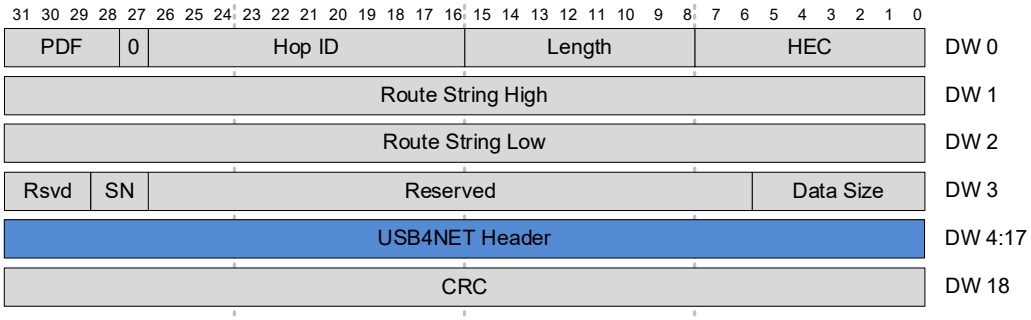
Table 5-6. Inter-Domain USB4NET Login Response Packet Fields

<table>
<tr><th>DW</th><th>Bits</th><th>Description</th></tr>
<tr><td>18</td><td>31:0</td><td>Status
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>Success</td></tr>
<tr><td>1 – 0x7FFFFFFFH</td><td>Reserved. Receiving any value in this range shall be considered a failure response to the corresponding USB4NET Login Request.</td></tr>
<tr><td>0x80000000H+</td><td>Vendor defined Failure</td></tr>
</table></td></tr>
<tr><td>19:20</td><td>31:0</td><td>Responder MAC Address – The responder’s MAC address. Responder MAC Address Hi is most significant 4 bytes of the responder’s MAC address.</td></tr>
<tr><td>21</td><td>31:0</td><td>Responder MAC Address Length – The length, in bytes, of the responder’s MAC address. Typically, 6.</td></tr>
</table>

5.1.2.4 Inter-Domain USB4NET Logout Request Packet

Figure 5-4 defines the structure of an Inter-Domain USB4NET Logout Request Packet.

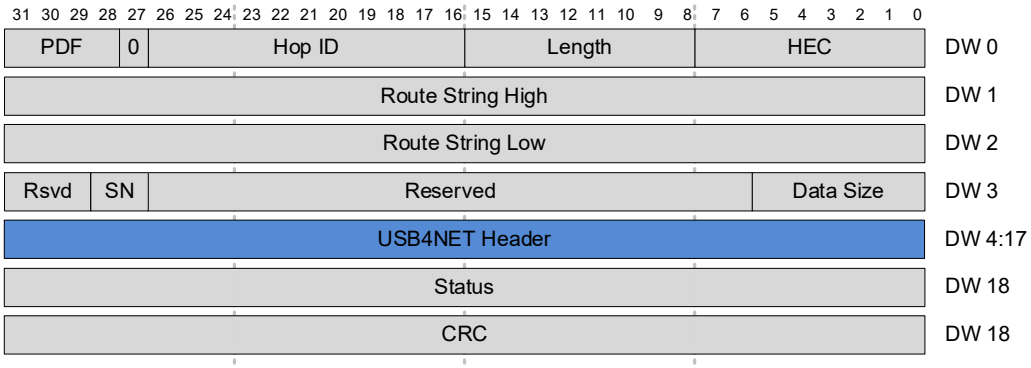
Figure 5-4. Inter-Domain USB4NET Logout Packet Format



5.1.2.5 Inter-Domain USB4NET Logout Response Packet

Figure 5-5 defines the structure of an Inter-Domain USB4NET Logout Response Packet.

Figure 5-5. Inter-Domain USB4NET Logout Response Packet Format



Refer to Table 5-6 for the definition of the *Status* field.

5.1.3 Tunneling Ethernet Packets over USB4

Once a Path has been established for USB4NET Service, Ethernet frames are tunneled across this Path. These frames are prepended with an USB4NET Frame header as described below.

Note: USB4NET does not directly handle any network packet retransmission. The network stack automatically attempts retransmission for any traffic (TCP packets) that is guaranteed delivery. For instance, TCP retransmits any TCP packet that fails or times out and USB4NET will dutifully (re)transmit it. UDP may not retransmit and its protocol is designed to accommodate this.

A Tunneled USB4NET Packet assigns a 1h value to the SoF field and a 2h value to the EoF field in the USB4 Packet Header. See the USB4 Specification for details on the SoF and EoF fields.

The USB4NET Frame Header shall be posted into a Data Buffer in host memory using Little Endian notation. Each field in the USB4NET Frame Header is posted such that the least-significant byte resides in the lowest-addressed byte in host memory.

The USB4NET Ethernet data shall be posted into a Data Buffer in host memory using Big Endian notation. An Ethernet frame depicted in Figure 5-6 in Ethernet network order is posted byte-wise from left to right, starting with the lowest-addressed byte in host memory.

5.1.3.1 Tunneling of Ethernet frames with up to 1500B of payload

A Connection Manager shall support transmitting and receiving Ethernet frames as described in this section.

Ethernet frames sent down from the network stack are packaged and pushed onto a Transmit Descriptor Ring, where they will be sent to the Inter-Domain peer. When a USB4 packet is received by the peer host, it is parsed, and the Ethernet frame is passed to the corresponding network stack.

Figure 5-6 depicts encapsulation of a single Ethernet frame into a USB4 Frame. The USB4 Frame starts with a 12-byte header defined in Table 5-7, followed by an Ethernet frame with up to 1518 bytes. Each Ethernet frame is encapsulated into a separate USB4 Frame.

Figure 5-6. Inter-Domain USB4NET Frame Format

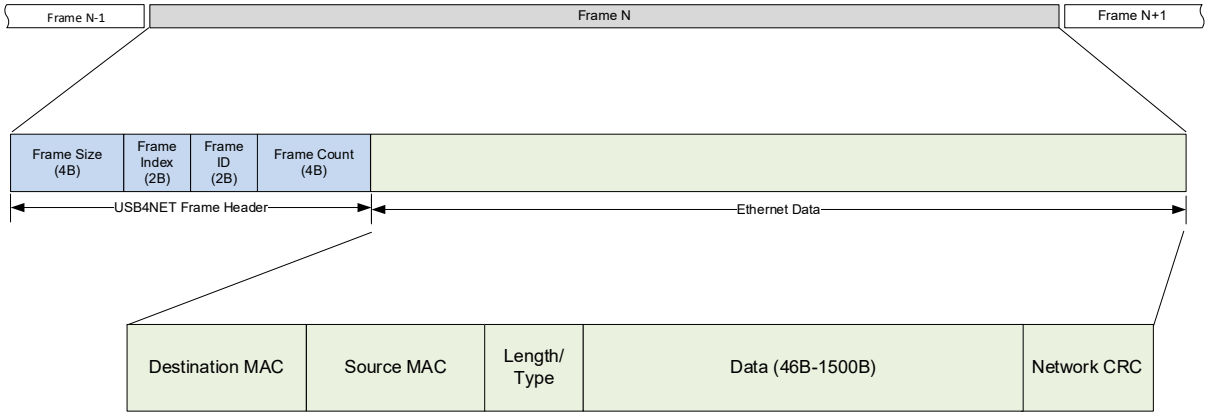


Table 5-7. Inter-Domain USB4NET Frame Format Fields

Field	Description
Frame Size (32 bits)	The length, in bytes, of this USB4 frame.
Frame Index (16 bits)	The n-th frame of this network packet (zero based). Shall be set to 0.
Frame ID (16 bits)	A sequential ID to distinguish this frame from the previous and next. This field continuously increments by one each time a frame is sent (with wraparound to zero when it reaches 0xFFFF).
Frame Count (32 bits)	The total number of frames that are expected for this network packet. Shall be set to 1.

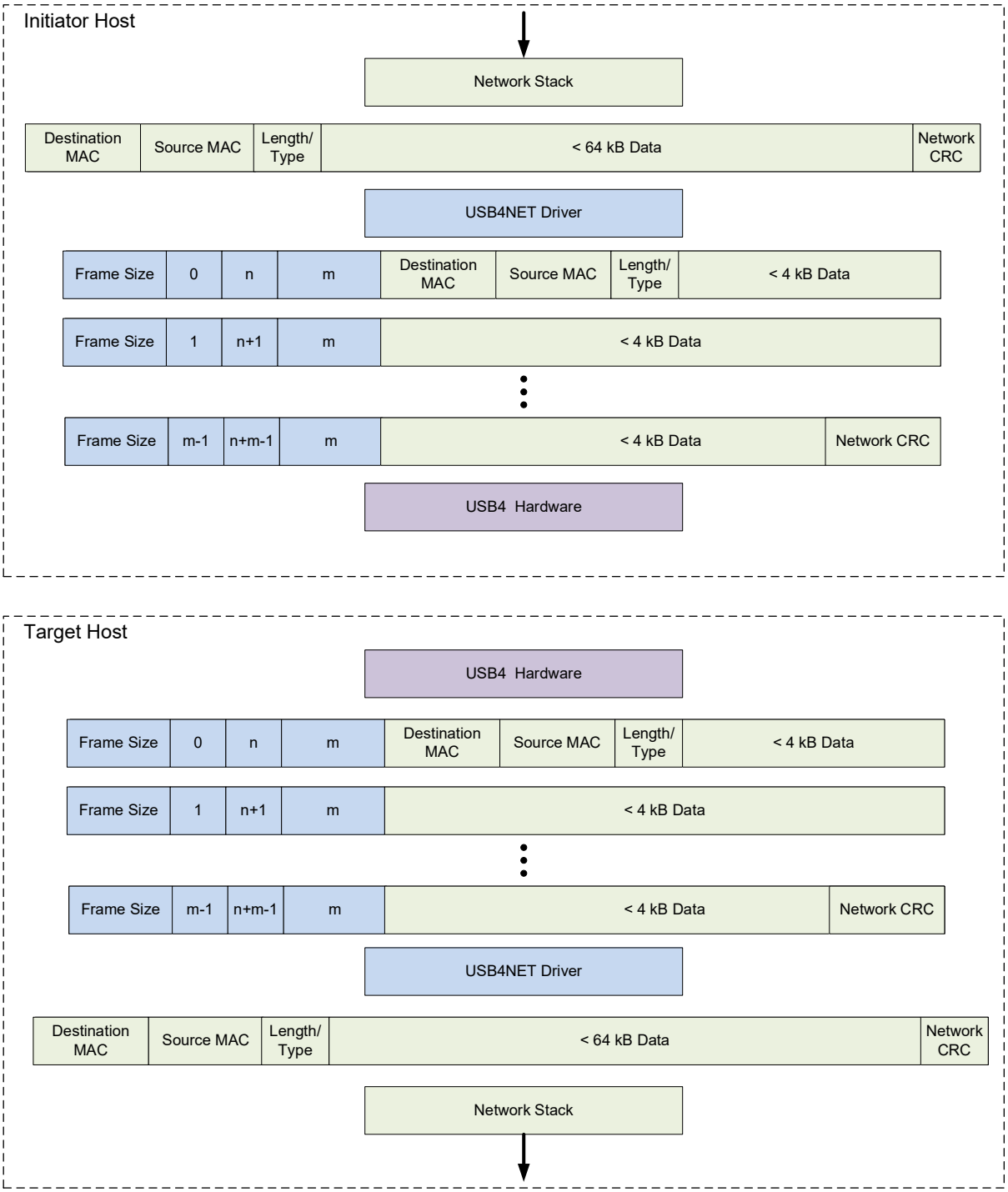
5.1.3.2 Tunneling of 64KB Ethernet Frames

A Connection Manager shall receive large Ethernet frames as described in this section. A Connection Manager may transmit large Ethernet frames as described in this section. However, if bit[2] in the *prtcstns* Immediate Value field (see Table 5-2) of the partner Connection Manager is set to 0b, it is recommended that the Connection Manager does not transmit large Ethernet frames.

USB4NET supports encapsulation of larger Ethernet frames with payloads of up to 64KB. As described in Figure 5-7, USB4NET software splits the large Ethernet frame into 4KB USB4 Frames (with a possible exception of the last USB4 Frame that may be shorter). Each 4KB Frame starts with a USB4NET Frame Header. The USB4 Frames are delivered to the Host Interface for transmission to the USB4 peer Domain.

The receiving Host Interface at the peer Domain delivers the received USB4 Frames to its USB4NET software. The receiving USB4 software reconstructs the original large Ethernet frame from the received USB4 Frames.

Figure 5-7. Encapsulation of 64KB Ethernet Frames



5.1.3.3 MAC Address

Higher level software running on each Host usually requires unique MAC Addresses to properly operate a Network stack. This section defines how the Local Connection Manager may generate one using the 64-bit *UUID* field in the Host Router's Router Configuration Space. The UUID is the basis for a MAC address-style identifier for USB4NET, being condensed to the appropriate size via the algorithm below.

```
uint32_t deviceID           = (UUID >> 4) & 0x0FFFFFFF;
uint16_t projectID_swapd    = bit_swap_16((UUID >> 32) & 0x0000FFFF);
```

```
uint16_t authorityID_swapd  = bit_swap_16((UUID >> 48) & 0x0000FFFF);
bool multicast              = false;
bool locallyAdministered    = true;

MACAddress.bytes[5] =  ((deviceID << 4) & 0xF0) |
                       ((routerID << 2) & 0x0C) |
                       (USB4PortNumber & 0x03);
MACAddress.bytes[4] =  ((deviceID >> 4) & 0xFF);
MACAddress.bytes[3] =  ((deviceID >> 12) & 0xFF);
MACAddress.bytes[2] =  ((deviceID >> 20) & 0xFF) ^
                       ((authorityID_swapd >> 0) & 0x03);
MACAddress.bytes[1] =  ((authorityID_swapd >> 2) & 0xFF);
MACAddress.bytes[0] =  (((multicast ? 1 : 0) << 0)) |
                       ((locallyAdministered ? 1 : 0) << 1) |
                       ((authorityID_swapd >> 8) & 0xFC) ^
                       ((projectID_swapd) & 0xFC);
```

5.1.3.4 Flow Control

Each Connection Manager shall enable per-Link flow control over the inter-Domain Link. A Dedicated Flow Control scheme shall be used.

If End-to-End (E2E) flow control is supported by both peers (as reported in their Properties Blocks), then each Connection Manager shall enable End-to-End flow control in both Transmit and Receive Descriptor Rings.

5.1.4 Example USB4NET Inter-Domain Properties in the Properties Block

Table 5-8. Inter-Domain USB4NET in a Properties Block

Offset	Value	Decoding
		Start of a Directory Data Offset entry
16	6e657477	"netw"
17	6f726b00	"ork"
18	44 00 0014	"D" R 20
19	00000021	Directory data offset, (Network Directory)
		Start of a Directory Data entry
21	ca8961c6	Directory UUID, Network Directory
22	9541ce1c	Directory UUID, Network Directory
23	5949b8bd	Directory UUID, Network Directory
24	4f5a5f2e	Directory UUID, Network Directory
25	70727463	"prtc"
26	69640000	"id"
27	76 00 0001	"v" R 1
28	00000001	Immediate value, Network Protocol ID
29	70727463	"prtc"
2A	76657273	"vers"
2B	76 00 0001	"v" R 1
2C	00000001	Immediate value, Network Protocol Version
2D	70727463	"prtc"
2E	72657673	"revs"
2F	76 00 0001	"v" R 1

Offset	Value	Decoding
30	00000001	Immediate value, USB4NET device driver Revision
31	70727463	"prtc"
32	73746e73	"stns"
33	76 00 0001	"v" R 1
34	00000003	Immediate value, Network Protocol Settings

A Example Inter-Domain Bonding Flow

Figure A-1and Figure A-2 give an example flow of the Bonding Flow.

Figure A-1. Bonding Flow – Part I

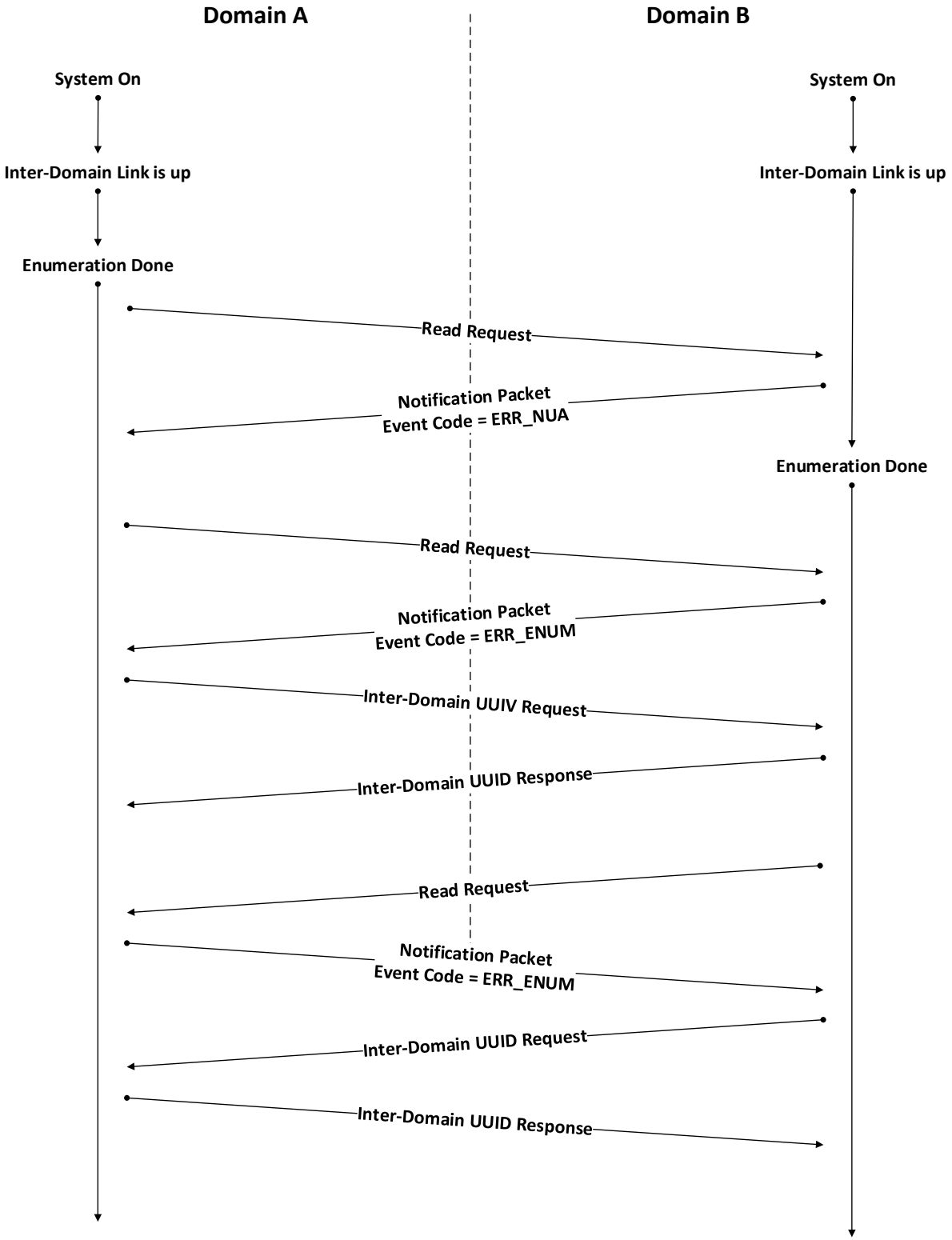


Figure A-2. Bonding Flow – Part II