Intel® Ethernet Controller E810

Dynamic Device Personalization (DDP) Technology Guide

Ethernet Products Group (EPG)

September 2020

Revision 2.1
617015-002

Revision History

Revision	Date	Comments
2.1	September 24, 2020	Updates include the following: • Updated OS-Default Package version from 1.3.13.0 to 1.3.16.0. • Updated DDP Comms Package version from 1.3.17.0 to 1.3.20.0. • Updated Step 2 in Section 4.3.6, "Loading a Specific DDP Package on a Specific 800 Series Network Adapter in Linux".
2.0[1]	July 23, 2020	Initial public release.

1. There are no previous publicly-available versions of this document.

Contents

NOTE: ***This page intentionally left blank.***

1.0 Introduction

Intel® Ethernet 800 Series (800 Series) is the next generation of Intel® Ethernet Controllers and Network Adapters. The Intel® Ethernet 800 Series is designed with an enhanced programmable pipeline, allowing deeper and more diverse protocol header processing. This on-chip capability is called Dynamic Device Personalization (DDP). Unlike the optional DDP solution in the Intel® Ethernet 700 Series (700 Series), the DDP implementation in the 800 Series is integral to the primary functions of the network packet processing pipeline. Similar to the 700 Series, enhanced DDP profiles can be loaded per device for specific capabilities. In the 800 Series, a DDP profile is loaded dynamically on driver load per device.

A general purpose DDP package is automatically installed with all supported 800 Series drivers on Windows, ESX, FreeBSD, and Linux operating systems, including those provided by the Data Plane Development Kit (DPDK). This general purpose DDP package is known as the OS-default package. Additional DDP packages will be available to address packet processing needs for specific market segments. For example, a telecommunications (Comms) DDP package has been developed to support GTP and PPPoE protocols in addition to the protocols in the OS-default package. The Comms DDP package is available with DPDK 19.11 and will also be supported by the 800 Series *ice* driver on Linux operating systems.

This document describes how the DDP packages are loaded or selected in various operating systems, the benefits of DDP features, and supported packet types in the OS-default DDP package. Also included are examples of DDP in use, including filters to direct packets to hardware queues.

2.0 How DDP Works

The 800 Series on-chip packet processing pipeline is shown in Figure 1. The DDP package programs functionality in both the parser and switch blocks in the pipeline, allowing dynamic support for new and existing protocols. Each of the subsequent lookup stages can also be configurable, forming a programmable packet processing pipeline. This pipeline can then handle packet identification, classification, and distribution in the network interface rather than in the OS, potentially offloading CPU cycles. Using this capability together with the 800 Series driver, host software can create filters to route specific packets to desired hardware queues for better CPU utilization.

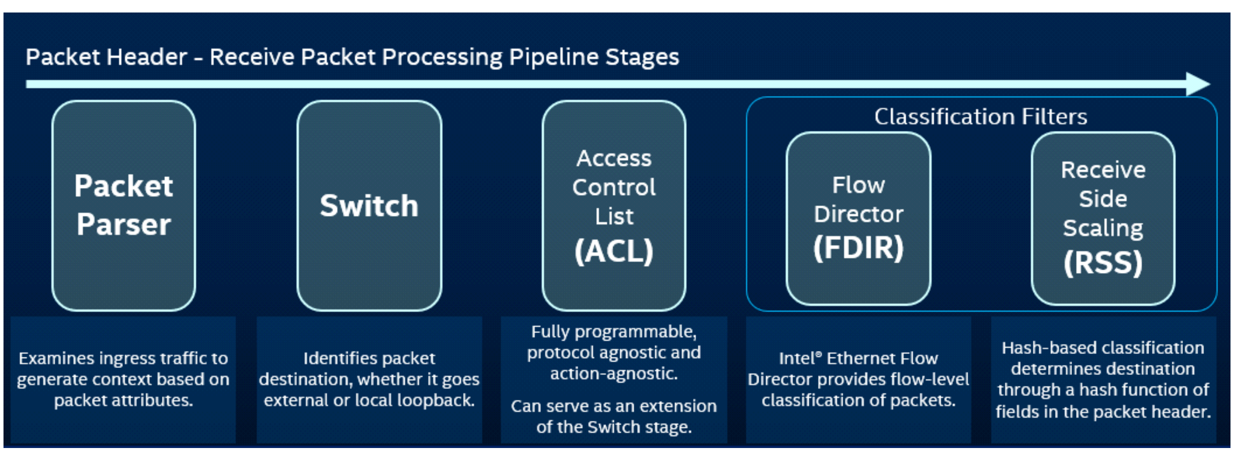

Figure 1. Intel® Ethernet 800 Series On-Chip Packet Processing Pipeline

DDP packages primarily contain static configuration information applied during device initialization. Furthermore, only one package can be active per device at a time. The device has a default Non-Volatile Memory (NVM) configuration that provides limited functionality for the system in pre-boot (before OS boot). During the OS boot process, the device driver loads the runtime DDP package that provides the more advanced capabilities of the packet processing pipeline.

Figure 2 shows the benefits of DDP protocol support by an example of processing a GTP Ethernet packet with and without the DDP package support for GTP protocols. With the OS-default package, which does not have GTP protocol support, the packet parser can only identify up to the first UDP header information. The result of identifying a packet as MAC_IPVx_UDP_PAY limits filtering for workload acceleration.

With GTP protocol support in the enhanced DDP Comms package, the GTP packet is fully identified (i.e., MAC_IPV4_GTPU_IPVx_TCP/UDP_PAY), enabling hardware filtering, accelerators, and RSS.

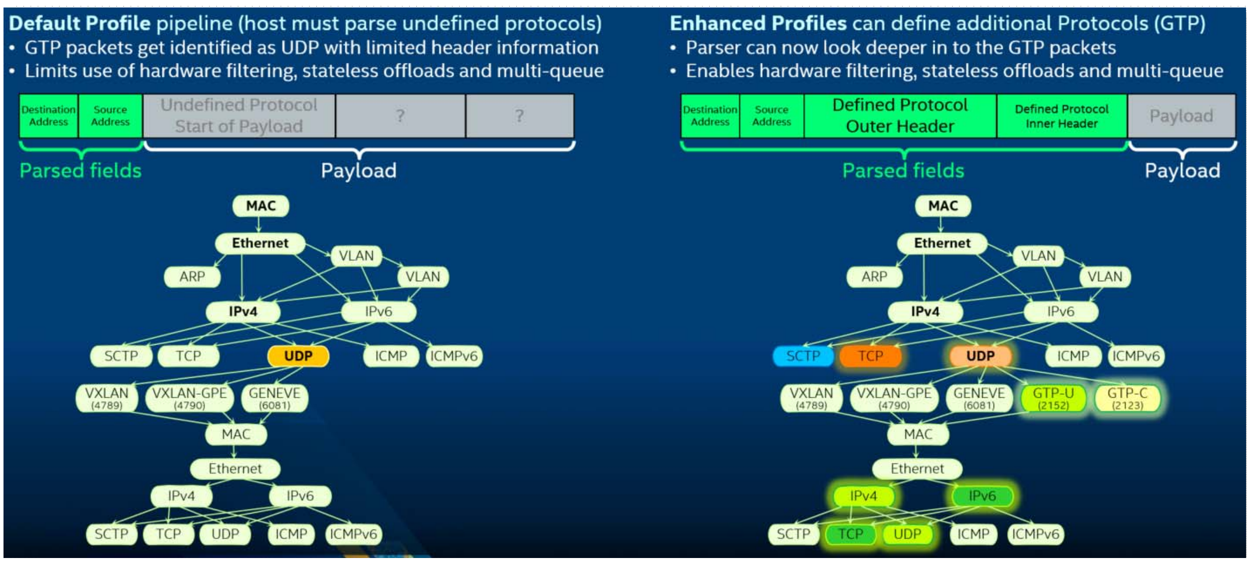

Figure 2. Benefits of Protocol Support by Enhanced DDP Package

3.0 Intel® Ethernet 800 Series Improvement over Previous Generations

The 800 Series incorporates many changes in hardware design to enhance packet processing capability. Table 1 shows key enhancements from the 700 Series to the 800 Series.

Table 1. DDP Enhancement Features

Feature	700 Series	800 Series
Maximum Receive Side Scaling (RSS) queues per physical function.	64	256
VSIs per device	384	768
Maximum unique packet types	192	1024
Intel® Ethernet FD Filters	Up to 8K	Up to 16K
Maximum packet header processing depth	256 bytes	Up to 504 bytes with up to 16 protocols deep

Table 1. DDP Enhancement Features [continued]

Feature	700 Series	800 Series
Custom DDP package loading	DPDK or i40e driver using **ethtool**	DPDK or 800 Series driver on startup
OS-default package loading	Default configuration is included as part of device NVM	DPDK or 800 Series driver on startup
Number of protocols supported per DDP package	One	Multiple
Different package selected per device	Yes	Yes

4.0 DDP Package Definitions and Usage

Figure 3 shows DDP configurations available at different stages of the system boot process in pre-boot and in OS boot.

In pre-boot or before a DDP package is loaded by an OS driver, an NVM-default configuration is automatically loaded by firmware. This configuration, referred to as "safe mode", supports a minimum set of protocols and allows basic packets handling in the pre-boot environment, such as PXE boot or UEFI. This NVM-default configuration is built into the device NVM firmware.

After OS boot, OS-default DDP, market-specific, or custom DDP package can be loaded by either the 800 Series driver or DPDK Configuration Tools.

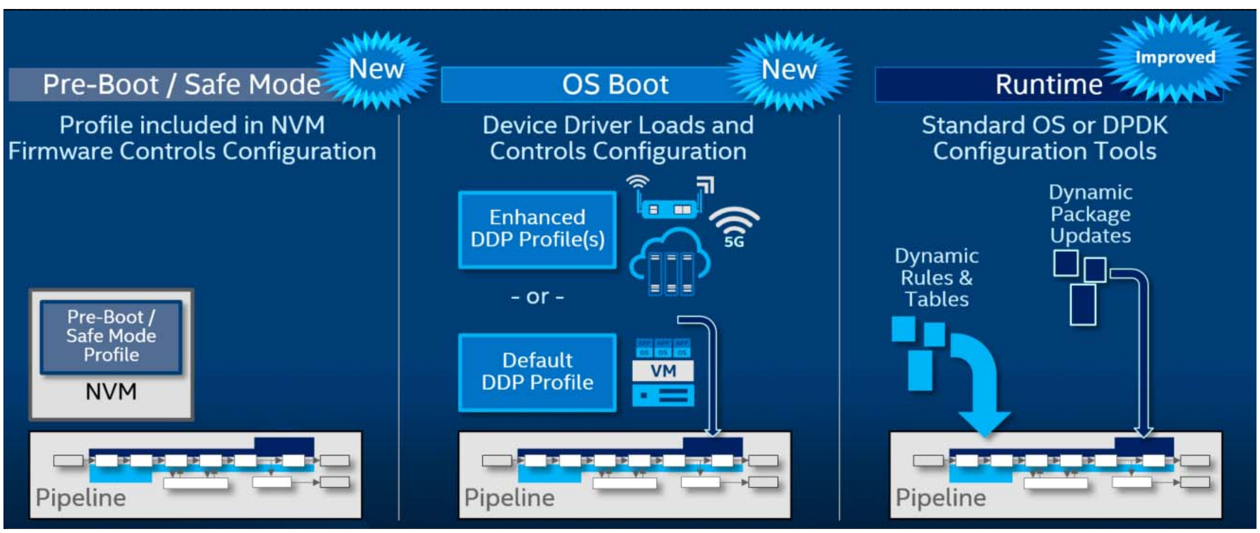

Figure 3. Intel® Ethernet 800 Series with Programmable Pipeline via DDP Profiles

4.1 Device Safe Mode (pre-Boot or without DDP Package)

In pre-boot or if a DDP package is not loaded by an OS driver, the 800 Series is configured in safe mode via an NVM-default configuration that is automatically loaded by firmware. This configuration supports a minimum set of protocols and allows basic packet handling in the pre-boot environment, such as PXE boot or UEFI.

The device can also be configured in safe mode if the DDP package fails to load due to a software incompatibility or other issue. If an OS driver loads and cannot load a DDP package, a message is printed in the system log that the device is now in safe mode.

In this safe mode, the driver disables support for the following features:

- Multi-queue
- Virtualization (SR-IOV/VMQ)
- Stateless workload acceleration for tunnel overlays (VxLAN/Geneve)
- RDMA (iWARP/RoCE)
- RSC
- RSS
- DCB /DCBx
- Intel® Ethernet Flow Director
- QinQ
- XDP / AF-XDP
- ADQ

Table 2 outlines the limited set of protocols supported in safe mode.

Table 2. Safe Mode Supported Protocols and Packet Types

Protocols	PTYPEs
MAC	MAC_PAY
ETYPE	MAC_LLDP
VLAN	MAC_ARP
IPv4	MAC_IPV4FRAG
IPv6	MAC_IPV4_PAY
TCP	MAC_IPV4_UDP_PAY
UDP	MAC_IPV4_TCP
SCTP	MAC_IPV4_SCTP
ICMP	MAC_IPV4_ICMP
ICMPv6	MAC_IPV6FRAG
LLDP	MAC_IPV6_PAY
ARP	MAC_IPV6_UDP_PAY
	MAC_IPV6_TCP
	MAC_IPV6_SCTP
	MAC_IPV6_ICMPv6

4.2 OS-Default Package

The OS-default DDP package is included with the 800 Series base driver. The DDP package is installed automatically when the driver is installed. It is loaded into the device during driver initialization in operating system boot. It is recommended to install the driver on a system running compatible device NVM firmware.

4.3 Driver Load of the OS-Default Package

4.3.1 VMware ESXi OS Driver Load of DDP Package

4.3.1.1 DDP Package Installation/Load

For VMware, the DDP package is compiled into the base driver and requires no additional installation or configuration for use.

4.3.1.2 Verifying DDP Package Status on VMware ESXi

The *vmkernel.log* file in */var/log/vmware/* shows the status of DDP package loading.

```
2019-11-20T10:49:49.748z cpu10:2098026) icen: icen_LogPkg:1976: 000:04:00.0: The DDP
package was successfully loaded: ICE OS Default Package version 1.3.16.0.
2019-11-20T10:49:49.860z cpu10:2098026) icen: icen_LogPkg:1976: 000:04:00.1: The DDP
package was successfully loaded: ICE OS Default Package version 1.3.16.0.
```

4.3.2 Windows OS Driver Load of DDP Package

4.3.2.1 DDP Package Installation/Load

For Windows operating systems, the DDP package is compiled into the base driver and requires no additional installation or configuration for use.

4.3.2.2 Verifying DDP Package Status on Windows

The status of package loading can be found under the System Logs.

Using the **Event Viewer** application, the status can be found under:

Event Viewer > **Windows Logs** > **System**

with *icea* as the source log.

Or, the status messages can be found using Powershell:

```
Get-WinEvent -provider icea | ? Message -like *DDP*
```

Following is an example of a Windows system event log shows successful loading of an OS-default package:

```
Intel(R) Ethernet Network Adapter E810-C-Q2 #2
The DDP package was successfully loaded: ICE OS Default Package version 1.3.16.0
```

4.3.3 Linux Driver Load of DDP Package

For Linux-based operating systems, the DDP package is included with the *ice* Linux base driver source code.

4.3.3.1 Out-of-Tree Driver DDP Installation

For the out-of-tree (OOT) driver, the driver installation process automatically installs *ice-x.x.x.x.pkg* to the */lib/firmware/updates/intel/ice/ddp* directory and creates a symbolic link to *ice.pkg*.

4.3.3.2 Upstream DDP Installation

Upstream *ice* drivers are supported on kernel.org v5.4 mainline kernel or higher. The required DDP package (and symbolic link) must be installed separately through downloading and installing version 20191022 or higher of the *linux-firmware* package from the following git repository:

https://git.kernel.org/pub/scm/linux/kernel/git/firmware/linux-firmware.git

4.3.3.3 DDP Package Driver Load

In either case (out-of-tree or upstream drivers), the *ice* driver looks for *ice.pkg* under the appropriate directory (depending on whether it is upstream or out of tree) and loads it during driver initialization.

The kernel message log (e.g., *dmesg*) indicates the status of package loading in the system. If the driver successfully finds and loads the DDP package, *dmesg* indicates that the DDP package is successfully loaded.

Following is an example of a *dmesg* indicating successful loading of the OS-default DDP package on a four-port device. The package is loaded by the first Physical Function (PF), and remaining PFs use the loaded DDP package.

```
# dmesg | grep -i ddp
ice 0000:82:00.0: The DDP package was successfully loaded: ICE OS Default Package
version 1.3.16.0
ice 0000:82:00.1: DDP package already present on device: ICE OS Default Package
version 1.3.16.0
ice 0000:82:00.2: DDP package already present on device: ICE OS Default Package
version 11.3.16.0
ice 0000:82:00.3: DDP package already present on device: ICE OS Default Package
version 1.3.16.0
```

4.3.4 FreeBSD Driver Load of DDP Package

For FreeBSD, the package is embedded into a firmware module and loaded by the PF driver using a native FreeBSD Firmware API for firmware updates. The firmware module is released with the base driver, and is installed by default with the driver package with no additional configuration necessary. *Dmesg* shows the loading status of the DDP package similar to Linux. The user can also use the **sysctl** command to view the loading status and the loaded package version.

```
# sysctl dev.ice.0.ddp_version
dev.ice.0.ddp_version: ICE OS Default Package version 1.3.16.0
```

4.3.5 DPDK Driver Load of DDP Package

If the system boots up with corresponding *ice* Linux base driver, the driver loads the DDP package as mentioned in the Linux DDP Driver load section. However, if the *ice* driver has not loaded a DDP package on the system at the time of the DPDK start, DPDK requires its own DDP installation process.

4.3.5.1 DPDK DDP Package Installation

If the DDP package has not been installed and loaded by the *ice* driver, DPDK requires a manual DDP package installation.

The user can download the DDP package from the Intel download center and extract the zip file to obtain the package (*.pkg*) file.

Similar to the Linux base driver, the DPDK driver looks for *intel/ice/ddp/ice.pkg* in the kernel default firmware search path */lib/firmware/updates* or */lib/firmware/*.

4.3.5.2 DPDK DDP Package Load

When the DPDK driver loads, it looks for *ice.pkg* to load in */lib/firmware/intel/ice/ddp/* or */lib/firmware/updates/intel/ice/ddp/*. If the file exists and it has not already been loaded, the driver downloads it into the device.

The kernel message log (e.g., *dmesg*) indicates the status of package loading in the system. If the driver successfully finds and loads the DDP package, dmesg indicates that the DDP package is successfully loaded.

Following is an example of a dmesg indicating successful loading of the Comms DDP package on a two-port device. The package is loaded by the first Physical Function (PF), and remaining PFs use the loaded DDP package.

```
# dmesg | grep -i ddp
ice 0000:3b:00.0: The DDP package was successfully loaded: ICE Comms Package version
1.3.20.0
ice 0000:3b:00.1: DDP package already present on device: ICE OS Comms Package version
1.3.20.0
```

DPDK's **testpmd** application also indicates the status and version of the loaded DDP package.

The example shows the **testpmd** output of a successful Comms package loading.

```
EAL: PCI device 0000:3b:00.1 on NUMA socket 0
EAL:   probe driver: 8086:1592 net_ice
ice_load_pkg_type(): Active package is: 1.3.20.0, ICE COMMS Package
```

4.3.6 Loading a Specific DDP Package on a Specific 800 Series Network Adapter in Linux

On a host system running with multiple 800 Series devices, there is sometimes a need to load a specific DDP package on a selected device while loading a different package on the remaining devices.

The 800 Series Linux base driver and DPDK driver can both load a specific DDP package to a selected adapter based on the device's serial number. The driver does this by looking for a specific symbolic link package filename containing the selected device's serial number.

The following example illustrates how a user can load a specific package (e.g., *ice-1.3.20.0.pkg*) on the device of Bus 6.

1. Find device serial number.

   To view bus, device, and function of all 800 Series Network Adapters in the system:

   ```
   # lspci | grep -i Ethernet | grep -i Intel
   06:00.0 Ethernet controller: Intel Corporation Ethernet Controller E810-C for QSFP
   (rev 01)
   06:00.1 Ethernet controller: Intel Corporation Ethernet Controller E810-C for QSFP
   (rev 01)
   82:00.0 Ethernet controller: Intel Corporation Ethernet Controller E810-C for SFP
   (rev 01)
   82:00.1 Ethernet controller: Intel Corporation Ethernet Controller E810-C for SFP
   (rev 01)
   82:00.2 Ethernet controller: Intel Corporation Ethernet Controller E810-C for SFP
   (rev 01)
   82:00.3 Ethernet controller: Intel Corporation Ethernet Controller E810-C for SFP
   (rev 01)
   ```

Use the **lspci** command to obtain the selected device serial number:

```
# lspci -vv -s 06:00.0 | grep -i Serial
Capabilities: [150 v1] Device Serial Number 35-11-a0-ff-ff-ca-05-68
```

Or, fully parsed without punctuation:

```
# lspci -vv -s 06:00.0 |grep Serial |awk '{print $7}'|sed s/-//g
3511a0ffffca0568
```

2. Rename the package file with the device serial number in the name.

   Copy the specific package over to */lib/firmware/updates/intel/ice/ddp* (or */lib/firmware/intel/ice/ddp*) and create a symbolic link with the serial number linking to the package, as shown. The specific symbolic link filename starts with “ice-” followed by the device serial in lower case without dash ('-').

```
# ln -s /lib/firmware/updates/intel/ice/ddp/ice-1.3.20.0.pkg /lib/firmware/
updates/intel/ice/ddp/ice-3511a0ffffca0568.pkg
```

   Or:

```
ln -sf /lib/firmware/updates/intel/ice/ddp/ice_comms-1.3.20.0.pkg ice-
e0680bffffb7a640.pkg
```

   Check softlink:

```
ll ice.pkg
lrwxrwxrwx. 1 root root 58 Sep 10 01:24 ice.pkg -> /lib/firmware/updates/intel/
ice/ddp/ice_comms-1.3.20.0.pkg
```

3. If using Linux kernel driver (*ice*), reload the base driver (not required if using only DPDK driver).

```
# rmmod ice
# modprobe ice
```

   The driver loads the specific package to the selected device and the OS-default package to the remaining 800 Series devices in the system.

4. Verify.

   For kernel driver:

   Following is an example of successful loading of the specific DDP package on the selected device of Bus 6 and OS-default package on the other device of Bus 82:

```
# dmesg | grep -i "ddp \| safe"
ice 0000:06:00.0: The DDP package was successfully loaded: ICE COMMS Package
version 1.3.20.0
ice 0000:06:00.1: DDP package already present on device: ICE COMMS Package version
1.3.20.0
ice 0000:82:00.0: The DDP package was successfully loaded: ICE OS Default Package
version 1.3.16.0
ice 0000:82:00.1: DDP package already present on device: ICE OS Default Package
version 1.3.16.0
ice 0000:82:00.2: DDP package already present on device: ICE OS Default Package
version 1.3.16.0
ice 0000:82:00.3: DDP package already present on device: ICE OS Default Package
version 1.3.16.0
```

   If DPDK is used:

   Verify using DPDK's **testpmd** application to indicate the status and version of the loaded DDP package.

4.4 OS-Default Package Protocol Support

Once the OS-default package is successfully loaded, the following protocols and packets are supported, as shown in Table 3 and Table 4.

Table 3. OS-Default DDP Package Supported Protocols

Protocols			
MAC	TCP	CTRL	GRE
ETYPE	UDP	LLDP	NVGRE
VLAN	SCTP	ARP	GENEVE
IPv4	ICMP	VXLAN-GPE	RoCEv2
IPv6	ICMPv6	VXLAN (non-GPE)	

Table 4. OS-Default DDP Package Supported Packet Types

PTYPE	PTYPE Description	PTYPE	PTYPE Description
1	MAC_PAY	46	MAC_IPV4_TUN_IPV4_UDP_PAY
2	MAC_PTP	48	MAC_IPV4_TUN_IPV4_TCP
11	MAC_ARP	49	MAC_IPV4_TUN_IPV4_SCTP
278	MAC_CONTROL	50	MAC_IPV4_TUN_IPV4_ICMP
22	MAC_IPV4_FRAG	110	MAC_IPV6_TUN_IPV4_FRAG
23	MAC_IPV4_PAY	111	MAC_IPV6_TUN_IPV4_PAY
24	MAC_IPV4_UDP_PAY	112	MAC_IPV6_TUN_IPV4_UDP_PAY
26	MAC_IPV4_TCP	114	MAC_IPV6_TUN_IPV4_TCP
27	MAC_IPV4_SCTP	115	MAC_IPV6_TUN_IPV4_SCTP
28	MAC_IPV4_ICMP	116	MAC_IPV6_TUN_IPV4_ICMP
88	MAC_IPV6_FRAG	51	MAC_IPV4_TUN_IPV6_FRAG
89	MAC_IPV6_PAY	52	MAC_IPV4_TUN_IPV6_PAY
90	MAC_IPV6_UDP_PAY	53	MAC_IPV4_TUN_IPV6_UDP_PAY
92	MAC_IPV6_TCP	55	MAC_IPV4_TUN_IPV6_TCP
93	MAC_IPV6_SCTP	56	MAC_IPV4_TUN_IPV6_SCTP
94	MAC_IPV6_ICMPV6	57	MAC_IPV4_TUN_IPV6_ICMPV6
29	MAC_IPV4_IPV4_FRAG	117	MAC_IPV6_TUN_IPV6_FRAG
30	MAC_IPV4_IPV4_PAY	118	MAC_IPV6_TUN_IPV6_PAY
31	MAC_IPV4_IPV4_UDP_PAY	119	MAC_IPV6_TUN_IPV6_UDP_PAY
33	MAC_IPV4_IPV4_TCP	121	MAC_IPV6_TUN_IPV6_TCP
34	MAC_IPV4_IPV4_SCTP	122	MAC_IPV6_TUN_IPV6_SCTP
35	MAC_IPV4_IPV4_ICMP	123	MAC_IPV6_TUN_IPV6_ICMPV6
95	MAC_IPV6_IPV4_FRAG	59	MAC_IPV4_TUN_MAC_IPV4_FRAG
96	MAC_IPV6_IPV4_PAY	60	MAC_IPV4_TUN_MAC_IPV4_PAY
97	MAC_IPV6_IPV4_UDP_PA	61	MAC_IPV4_TUN_MAC_IPV4_UDP_PAY
99	MAC_IPV6_IPV4_TCP	63	MAC_IPV4_TUN_MAC_IPV4_TCP

Table 4. OS-Default DDP Package Supported Packet Types [continued]

PTYPE	PTYPE Description	PTYPE	PTYPE Description
100	MAC_IPV6_IPV4_SCTP	64	MAC_IPV4_TUN_MAC_IPV4_SCTP
101	MAC_IPV6_IPV4_ICMP	65	MAC_IPV4_TUN_MAC_IPV4_ICMP
36	MAC_IPV4_IPV6_FRAG	125	MAC_IPV6_TUN_MAC_IPV4_FRAG
37	MAC_IPV4_IPV6_PAY	126	MAC_IPV6_TUN_MAC_IPV4_PAY
38	MAC_IPV4_IPV6_UDP_PAY	127	MAC_IPV6_TUN_MAC_IPV4_UDP_PAY
40	MAC_IPV4_IPV6_TCP	129	MAC_IPV6_TUN_MAC_IPV4_TCP
41	MAC_IPV4_IPV6_SCTP	130	MAC_IPV6_TUN_MAC_IPV4_SCTP
42	MAC_IPV4_IPV6_ICMPV6	131	MAC_IPV6_TUN_MAC_IPV4_ICMP
102	MAC_IPV6_IPV6_FRAG	66	MAC_IPV4_TUN_MAC_IPV6_FRAG
103	MAC_IPV6_IPV6_PAY	67	MAC_IPV4_TUN_MAC_IPV6_PAY
104	MAC_IPV6_IPV6_UDP_PAY	68	MAC_IPV4_TUN_MAC_IPV6_UDP_PAY
106	MAC_IPV6_IPV6_TCP	70	MAC_IPV4_TUN_MAC_IPV6_TCP
107	MAC_IPV6_IPV6_SCTP	71	MAC_IPV4_TUN_MAC_IPV6_SCTP
108	MAC_IPV6_IPV6_ICMPV6	72	MAC_IPV4_TUN_MAC_IPV6_SCTP
43	MAC_IPV4_TUN_PAY	132	MAC_IPV6_TUN_MAC_IPV6_FRAG
58	MAC_IPV4_TUN_MAC_PAY	133	MAC_IPV6_TUN_MAC_IPV6_PAY
109	MAC_IPV6_TUN_PAY	134	MAC_IPV6_TUN_MAC_IPV6_UDP_PAY
124	MAC_IPV6_TUN_MAC_PAY	136	MAC_IPV6_TUN_MAC_IPV6_TCP
44	MAC_IPV4_TUN_IPV4_FRAG	137	MAC_IPV6_TUN_MAC_IPV6_SCTP
45	MAC_IPV4_TUN_IPV4_PAY	138	MAC_IPV6_TUN_MAC_IPV6_ICMPV6

NOTE: ***This page intentionally left blank.***

LEGAL

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

This document (and any related software) is Intel copyrighted material, and your use is governed by the express license under which it is provided to you. Unless the license provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this document (and related materials) without Intel's prior written permission. This document (and related materials) is provided as is, with no express or implied warranties, other than those that are expressly stated in the license.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors which may cause deviations from published specifications.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

Other names and brands may be claimed as the property of others.

© 2020 Intel Corporation.