PA Low Level Driver

Release Notes

Applies to Product Release: 03.00.02.07

Publication Date: Sep 21, 2018

Contributors to this document

Texas Instruments,
Incorporated
20250 Century Boulevard

Contents

Release Notes

PA Low Level Driver for 03.00.02.07

Overview

This document provides the release information for the latest PA LLD which should be used by drivers and application that interface with PA.

PA LLD module includes:

- Pre-compiled library for DSP (Big and Little) Endian of PA Low Level Driver.
- Sources, examples and unit test code.
- API reference guide

LLD Dependencies

- This release of PA LLD requires CSL package released with PDK
- RM LLD

Resolved Incident Reports (IR)

Table 2 provides information on IR resolutions incorporated into this release.

Table 2 Resolved IRs for this Release

IR Parent/ Child Number	Severity Level	IR Description

Known Issues/Limitations

Table 3 Known Issue IRs for this Release

IR Parent/ Child Number	Severity Level	IR Description
108274	Minor	PA examples use hard coded queue numbers

New/Updated Features and Quality

Release 3.0.2.07

- Update test config files to use ti.trace.Sysmin

Release 3.0.2.06

This is an **engineering release,** tested by the development team.

- Updated buildlib.xs to include Rules.make from ti/build infrastructure.
- Resolved IRs listed at section “Resolved Incident Reports (IR)”
- Firmware versions used in the release:

Description	Version
Pa firmware version for Gen1 devices (K2H, K2K, C6678)	**3.0.2.3**
Pa firmware version for Gen2 devices (K2L/K2E)	3.0.2.3

Release 3.0.2.05

This is an **engineering release,** tested by the development team.

- Resolved IRs listed at section “Resolved Incident Reports (IR)”

IR Parent/ Child Number	Severity Level	IR Description
PRSDK-3155	Major	PA_emacExample_BiosExampleProject ARM-LE: Timeout waiting for reply from PA to Pa_addMac command on K2 platforms

- Firmware versions used in the release:

Description	Version
Pa firmware version for Gen1 devices (K2H, K2K, C6678)	**3.0.2.3**
Pa firmware version for Gen2 devices (K2L/K2E)	3.0.2.3

Release 3.0.2.04

This is an **engineering release,** tested by the development team.

- Resolved IRs listed as below.

IR Parent/ Child Number	Severity Level	IR Description
PRSDK-2350	Major	Pa Classify1 on NetCP 1.0 hang with next packet header that is invalid to that PDSP.

- Firmware versions used in the release:

Description	Version
Pa firmware version for Gen1 devices (K2H, K2K, C6678)	**3.0.2.3**
Pa firmware version for Gen2 devices (K2L/K2E)	3.0.2.3

Release 3.0.2.03

This is an **engineering release,** tested by the development team.

- Resolved IRs listed at section “Resolved Incident Reports (IR)”
- Firmware versions used in the release:

Description	Version
Pa firmware version for Gen1 devices (K2H, K2K, C6678)	**3.0.2.2**
Pa firmware version for Gen2 devices (K2L/K2E)	**3.0.2.3**

- **Resolved IRs listed as below:**

IR Parent/ Child Number	Severity Level	IR Description
PRSDK-1881	Major	Support to ignore EQoS operations as long as the pa_CMD_REPORT_TX_TIMESTAMP is included.

Release 3.0.2.02

This is an **engineering release,** tested by the development team.

- Destination Queue Bounce Operation

  The PA LLD/Firmware is enhanced to support the Destination Queue Bounce Operation as part of the solution to deal with the potentialcache coherency deficiency identified at the Keystone2 devices where memory consistency is not guaranteed for IO coherent A15 and PktDMA masters at some rare conditions. Therefore it is possible that the data arrival signal to the ARM (i.e., presence of a descriptor in QMSS queue) may occur prior to the data arriving properly in the ARM cache. Thus, the ARM core may access stale data.

  To ensure ARM Cache consistency, one of the QMSS PDSP is enhanced to provide a DMA barrier function. Packets destined to ARM queues may be delivered first to one of the QMSS "bounce" queues serviced by this function. The QMSS PDSP f/w will pop packets from these queues, perform the necessary barrier operation (that will cause the ARM cache to get invalidated for the descriptor and buffer locations), and then will relay the packet to the final destination queue.

  The PASS Destination Queue Bounce Operation is designed to enable the QMSS proxy bounce on packets PA sends to queues served by ARM user space by embedding 2 control bits into the destination queue ID to instruct the PASS firmware to re-route the packets to the specified QMSS bounce queues.

  This operation can be enabled and configured by a new global configuration message including the following parameters:

  - Enable/Disable
  - QMSS Bounce Queue IDs
    - DDR Queue: All PktMDA descriptors and buffers use DDR memory only
    - MSMC Queue: PktDMA descriptor and buffers may use MSMC memory and/or DDR memory
  - NetCP hardware queue info
    - First NSS hardware Queue
    - Last NSS hardware Queue

- Default Behavior map []: Specify the default queue bounce operation for each traffic class such as Command response and QoS operation

The instance size of PA LLD is increased to 320 bytes to support this feature.

- Resolved IRs listed at section as below:

IR Parent/ Child Number	Severity Level	IR Description
PRSDK-1541	Major	PA LLD enhancement to support PKTDMA coherency workaround.

- **Firmware versions used in this release**

Description	Version
Pa firmware version for Gen1 devices (K2H, K2K, C6678)	3.0.2.1
Pa firmware version for Gen2 devices (K2L/K2E)	3.0.2.2

Release 3.0.2.01

This is an **engineering release,** tested by the development team.

Firmware versions used in this release

Description	Version
Pa firmware version for Gen1 devices (K2H, K2K, C6678)	3.0.2.0
Pa firmware version for Gen2 devices (K2L/K2E)	3.0.2.**1**

Resolved IRs listed as below:

IR Parent/ Child Number	Severity Level	IR Description
PRSDK-1313	Major	Pa Firmware for NetCp 1.5 (Gen2 – K2L/K2E): Fixed bug in firmware when L2-Capture debug feature is enabled.
PRSDK-1314	Major	Pa LLD bug fix for NetCp 1.5 (Gen 2 – K2L/K2E) when ACL remove entry is initiated

IR Parent/ Child Number	Severity Level	IR Description
PRSDK-1508	Major	Pa_addCustomLUT1 causes segmentation fault on NetCp 1.5 (Gen2 – K2L/K2E) devices.

Release 3.0.2.00

This is an **engineering release,** tested by the development team.

Starting from this release, Pa firmware versions would have independent versions from LLD and would be tracked in the table format as below. The goal is to update the individual firmware versions only if they have any changes/updates.

Firmware versions used in this release

Description	Version
Pa firmware version for Gen1 devices (K2H, K2K, C6678)	3.0.2.0
Pa firmware version for Gen2 devices (K2L/K2E)	3.0.2.0

Resolved IRs listed as below:

IR Parent/ Child Number	Severity Level	IR Description
PRSDK-971	Major	New PA firmware format to support use firmware upgrade from file system
PRSDK-963	Major	pa_RESUBMIT_COMMAND endianess bug

Release 3.0.1.18

This is an **engineering release,** tested by the development team

Resolved IRs listed as below.

IR Parent/ Child Number	Severity Level	IR Description
PRSDK-660	Major	PA does not read the CRC_disable flag from config command set configuration

IR Parent/ Child Number	Severity Level	IR Description

Release 3.0.1.17

This is an **engineering release,** tested by the development team

Resolved IRs listed as below.

IR Parent/ Child Number	Severity Level	IR Description
PRSDK-594	Major	After sending a malformed packet (IPv6 with no next header) Linux is unable to receive any packets, cannot send icmp ping
PRSDK-613	Major	C1 number of invalid states counter incrementing

Release 3.0.1.8

This is an **engineering release,** tested by the development team

Resolved IRs listed as below.

IR Parent/ Child Number	Severity Level	IR Description
120788	Major	After sending a malformed packet (IPv6 with no next header) Linux is unable to receive any packets, cannot send icmp ping
116671	Major	PA 1.3.1 Egress PDSP CRC Check Lockup
119264	Minor	PA_emacExample*ExampleProject fail intermittently on K2E platform for BE and LE
119889	Minor	PA: LLD needs to return new error code when LUT2 is full

Release 3.0.1.7

This is an **engineering release,** tested by the development team

Resolved IRs as listed below:

IR Parent/ Child Number	Severity Level	IR Description
120212	Major	PA: Untagged packet matching support

Release 3.0.1.6

- New API to get the virtual link ID from the handle is added.
- Support for building ARM RTOS libraries added

Resolved IRs as listed below:

IR Parent/ Child Number	Severity Level	IR Description
120065	Minor	PA: support Pa_getVirtualLinkId API

Release 3.0.1.5

- Updated serdes initialization sequence in cpsw_mgmt.c
- Support full size ACL LUT for NSS_GEN2 only

  Enabled support to utilize full size LUT for ACL, this feature enhancement triggered API **change** in the way Pa_addAcl() as below. The application needs to pass a new argument, pointer to a 32 bit value, as highlighted below. When this value is not zero, it flags the application that the Pa_addAcl() call is not successful by returning "*pa_ACL_BUSY*". The application can attempt again to add the ACL after "*timeToNextCall*" micro seconds time is elapsed.

```
paReturn_t  Pa_addAcl  (Pa_Handle        iHandle,
                        int              aclInst,
                        int              aclAction,
                        paAclInfo_t      *aclInfo,
                        paHandleL2L3_t   prevLink,
                        paHandleAcl_t    nextEntry,
                        paHandleAcl_t    *retHandle,
                        paCmd_t          cmd,
                        uint16_t         *cmdSize,
                        paCmdReply_t     *reply,
                        int              *cmdDest,
                        uint32_t         *timeToNextCall);
```

  In order to minimize the "*pa_ACL_BUSY*", if application has prior knowledge about how the new ACL entry is going to be inserted, a new parameter "*insertMode*" is added in "*paAclConfig_t*", which can take any of the below values. The default value is "*pa_ACL_INSERT_RANDOM*".

  pa_ACL_INSERT_TOP - Application adds new ACL entry to the top of the ACL table typically (typically new entry that is going to be inserted has highest priority).

  pa_ACL_INSERT_BOTTOM - Application adds new ACL entry to the bottom of the ACL table typically (typically new entry that is going to be inserted has lowest priority).

pa_ACL_INSERT_RANDOM - Application adds new ACL entry in any order (application has no prior knowledge on the priority of the new entries that are going to be inserted).

- GTP-U End marker message handling
    - The new feature supports option to treat GTP-U end marker packets as G-PDU packets.
- SPI with Link handling
    - Support SPI only with link for Gen 1 devices. Please note that when IpInfo has only SPI, previous link parameter is recommended to be set for Gen1 and mandatory for Gen2 devices due to hardware limitations.
- **Migration Information**
    - *"Pa_startCfg"* API return value changed to indicate whether the start is successful or not.
    - *No migration is needed if application is not using ACL for IP firewall operations. Here is the migration information for applications using ACL for IP firewall operations. Please refer to above section for details on Pa_addAcl() changes.*
        - *"pa_BUF_ACL_TABLE"* size requirement increased by (nMaxAcl * 20 + 24) bytes.
        - *Application modification is needed as"*Pa_addAcl()" API is breaking backwards compatibility.
- Resolved IRs as listed below:

IR Parent/Child Number	Severity Level	IR Description
SDOCM00116898	Major	PA: SPI with Link does not work
SDOCM00116499	Major	PA 1.5 firmware locks up for MTU size = 46 packets
SDOCM00115728	Major	PASS: Support IP forwarding with more than two layer of IPs
SDOCM00115449	Major	PASS does not process GRE packets on NSS Gen2 device correctly
SDOCM00115444	Major	PA LLD: PASS sub system address NULL checks in the LLD are missing
SDOCM00115345	Major	PASS: IP reassembly timeout at the inner RA causes the RA engine to hang
SDOCM00115334	Major	PA LLD: Eoam Mode Target flow classification statistics does not reflect the correct statistics
SDOCM00115312	Major	PASS: Tx CRC command does not work at K2L/K2E devices
SDOCM00115088	Minor	PA user space shared object libraries to have same shared object lib name (SONAME) across devices supported
SDOCM00114984	Major	Enhance Ethernet OAM unit Test to cover described test cases
SDOCM00114983	Major	PA LLD: Support for IPSec NAT-T detector enable feature along with EOAM feature
SDOCM00114982	Major	PA LLD support Packet Capture along with Ethernet OAM feature

SDOCM00114971	Major	PA Sub System: ACL LUTs (In0-PDSP1, In3-PDSP0) size currently limited to half size 128
SDOCM00114416	Major	GTP-U End Marker message handling
SDOCM00116833	Major	Protection for bad Tx Cmd entering the PASS, which can potentially hang the system
SDOCM00117285	Major	PA FW does not send WCDMA FP fragmented packets on K2L
SDOCM00117286	Major	PA FW mishandles WCDMA CRC on K2L
SDOCM00117710	Major	PA LLD to support Ethernet Traffic Forwarding through QoS
SDOCM00117941	Major	CPSW_LOOPBACK_NONE configuration sequence does not work

Release 3.0.1.4

The PA LLD NSS Gen2 is enhanced to support Ethernet OAM (EOAM) mode, which includes the EOAM classification and new packet flow. Note: PA LLD NSS Gen1 does not support this feature.

The Ingress0 PDSP1 LUT1 is utilized to support Ethernet OAM (EOAM) target flow classification instead of the firewall of outer IP and Ingress 3 PDSP0 is enhanced to filter both outer IP/UDP and inner IP/UDP.

This release is backward compatible to PA 3.0.1.3 with EOAM by default being in disabled mode with few exceptions as indicated below. Once EOAM is enabled, the PA Sub System (PASS) transitions to a new mode of operations not compatible with original PASS operations. Pa_create() API being called during initialization time supports enabling the EOAM feature. Refer doxygen documentation for details.

- **Migration Information**
    a. PA instance memory increased to 288 bytes from 256 bytes.
    b. PA Number of buffer requirement increased to 8 buffers from 7 buffers
    c. If new feature EOAM mode is enabled, note following features will not be supported
        - outer firewall (ACL) operations
        - Plain IP Tunnel (IP over IP) without IPSec packets

Note: Application modification may be required for a) and b).

Additional details for new feature and API changes in this release are described below:

- **Ethernet OAM (EOAM) Mode Operation**

    During EOAM classification, packets are inspected for a specific target flow match based on any group of Destination MAC address, Source MAC address, VLAN priority, VLAN ID and Ethernet PORT id. Each target flow is associated with the corresponding "user defined statistics counter". During the match if the message type/PDU found to be 1DM/DMM/DMR/LMM/LMR, PA LLD supports configurations to forward the packet to a host queue.
    - For IPSec transport mode, Ingress 3 would filter the IP/UDP header

- For IPSec tunnel mode, Ingress 3 would filter the inner IP/UDP header
- For a non-cipher packet, Ingress 3 would filter the IP/UDP header

Please refer to API header file "*ti/drv/pa/pa.h*" or *appendix 8* in Doxygen for detailed operations.

Additional API details
The EOAM global configuration structure is added to existing system configuration structure "*paSysConfig_t*" to configure the EOAM operation. The configuration is used to perform EOAM related configurations.
Additionally, "*Pa_control()"* API is enhanced to support to set timing offset in IEEE 1588 timestamp format for seconds and nano seconds.
Pa_TxCmd is enhanced to support patching of message with time in IEEE 1588 time format in the message and/or with receive packet counter at a specified offsets in the message. This is done using the commands "*pa_CMD_PATCH_TIME*" and "*pa_CMD_PATCH_COUNT*" in conjunction with the next route command.

Application code should follow the following procedure to configure and enable this feature.

- Initialize PA with EOAM buffer requirements
- Allocate 32bit user defined counters per EOAM target flow
- Global configuration enabling EOAM
- Setup Eoam target flow using "*Pa_addEoamFlow()*" API along with other L2, L3, Inner ACL and L4 configurations
- Initiate Pa_control() for any timing offset correction any time after EOAM global configuration is done in the operation.

- Resolved IRs as listed below.

IR Parent/ Child Number	Severity Level	IR Description
115006	Major	PA LLD: Support for Ethernet OAM feature

Release 3.0.1.3

This is an **engineering release,** tested by the development team

- Resolved IRs as listed as below.

IR Parent/ Child Number	Severity Level	IR Description
114514	Major	PASS: CRC calculation for large-size packet doesn't work on K2E/K2L devices
114515	Major	PASS: Fail to match default (catch-up) IP rule

IR Parent/ Child Number	Severity Level	IR Description
114543	Major	PASS cannot handle some illegal IP packets
114576	Major	PASS: Illegal Tx commands may cause infinite loop
114668	Major	PASS: Provide LUT1 workaround for ACL operation
114669	Major	PASS: Support SCTP port filter in ACL function

Release 3.0.1.2

- This release provides a mechanism to support post-classification L2 packet capture by adding a new parameter ctrlBitMap to paRouteInfo2_t for PASS gen2 devices. To enable post-classification packet capture, the application should configure the paRouteInfo2_t as the followings:
    - validBitMap: add pa_ROUTE_INFO_VALID_CTRLBITMAP
    - dest: pa_DEST_CONTINUE_PARSE_LUT1 or pa_DEST_CASCADED_FORWARDING_LUT1
    - flowId: CPPI flow ID to deliver captured packets
    - queue: destination queue of the captured packets
    - swInfo0: placed in SwInfo0 for captured packets
    - ctrlBitMap: pa_ROUTE_INFO_L2_PKT_CAPTURE
- Resolved IRs as listed below.

IR Parent/ Child Number	Severity Level	IR Description
114358	Major	PASS: User stats should be incremented at the IP forwarding path as well
114359	Major	PASS should be able to remove the outer IP header and trailer of forwarding IP packet
114360	Major	PASS: ACL byte count should include the associated IP length only
114361	Major	PASS: Enhance Egress Flow L2 processing to be able to forward packet to QoS queue with EMAC port number
114362	Major	PASS: Support Post-classification L2 packet capture on K2L/E devices
114374	Major	PASS: The ACL entry with DSCP does not work

Release 3.0.1.1

- This release expands the 3.0.1.0 feature supports to all keystone2 devices including K2L and K2E.
- This release includes a PASS feature enhancement which increases the size of PA timestamp from 48-bit to 64-bit to avoid the potential rollover issue. The lower 32-bit value of the 64-bit timestamp still resides at the timestamp field of the EPIB section of the CPPI host descriptor and the upper 32-bit timestamp value is moved from the last 16-bit value of 24-byte PASS Long Info section to the 7th 32-bit word of 32-byte PASS Long info as shown at the table below:

Section	offset	Description
EPIB	0	Timestamp (LSW: lower 32-bit)
	4	swInfo0
	8	swInfo1
	12	Swinfo2 (not used)
PASS LongInfo	0	PASS-specific
	4	PASS-specific
	8	PASS-specific
	12	PASS-specific
	16	PASS-specific
	20	PASS-specific
	24	Timestamp (MSW: upper 32-bit)
	28	Reserved

The timestamp MSB field extract macro PASAHO_LINFO_READ_TSTAMP_MSB(x) is redefined to reflect this change at pasahost.h.

The new PASS Long Info format defined above is applicable to all known applications running on DSP, ARM user mode and Linux Kernel except for the application which invokes the PA command pa_CMD_COPY_DATA_TO_PSINFO to copy some payload data to the PASS Long Info area byte24-byte31. In this case, the size of PASS Long Info will be increased from 32 to 40 bytes as shown below:

Section	offset	Description
EPIB	0	Timestamp (LSW: lower 32-bit)
	4	swInfo0
	8	swInfo1
	12	Swinfo2 (not used)
PASSS LongInfo	0	PASS-specific
	4	PASS-specific
	8	PASS-specific (or payload Info)
	12	PASS-specific (or payload Info)
	16	PASS-specific (or payload Info)
	20	PASS-specific (or payload Info)
	24	Payload Info
	28	Payload Info
	32	Timestamp (MSW: upper 32-bit)

	36	Reserved

.

A new macro PASAHO_LINFO_READ_TSTAMP_MSB2(x) is provided to extract the upper 32-bit timestamp in this use case.

In the egress direction, the TX timestamp report descriptor contains the lower 32-bit timestamp at the timestamp field and the upper 32-bit timestamp at the swInfo1 field within its EPIB section.

Besides, the PASS system timestamp data structure is enhanced to report the most significant 16-bit timestamp value at parameter hi_hi. This is to maintain backward compatibility with 48-bit timestamp.

For devices with the second generation PASS, the PA timestamp is replaced with the CPTS timestamp from the CPSW in the ingress direction. The pa_CMD_NEXT_ROUTE is also enhanced to support Tx timestamp request. To instruct CPSW (switch) to report the packet transmit timestamp as a CPTS event, the application should perform the following actions.

- Format tx command pa_CMD_NEXT_ROUTE with pa_NEXT_ROUTE_RPT_TX_TIMESTAMP bit set.
- Invoke macro a_FORMAT_REPORT_TIMESTAMP_INFO(domain, msgType, seqId) to format the timestamp control word at swInfo0 field of the CPPI descriptor.

- Resolved IRs as listed below

IR Parent/ Child Number	Severity Level	IR Description
113125	Major	PA LLD: Pa_addAcl does not support the same entry at both outer and inner ACL
113142	Major	PASS Firmware: Ingress packet sideband access can be to prior packet on PA so that packet will be delivered to incorrect destination
113996	Major	Modify PA firmware to return 64 bit timestamp counter values

Release 3.0.1.0

Release is backward compatible to PA 3.0.0.10 with following exception.
The classify1 PDSP image is divided into three images running on PDSP0, PDSP1 and PDSP2 respectively for enabling new features on NSS Gen1 devices. In the case of Linux Kernel downloading firmware change would need to be reflected. This will be available in default Kernel release from TI.

Additional details for new feature and API changes in this release are described below:

- EMAC Interface-based Packet Capture (Mirroring)

The ingress packet and/or the egress packet can be duplicated and forwarded to desired host queue or mirror port per interface configuration during first stage of ingress processing and the egress processing at PDSP. To minimize the performance impact when this feature is not enabled, there is a system-level global control in addition to the per interface configuration as described below:

- Additional two control bit definitions are available at validBitMap and ctrlBitMap within paPacketControl2Config_t to enable/disable the system-wide packet capture (mirroring) operation.
  - pa_PKT_CTRL_EMAC_IF_IGRESS_CLONE
  - pa_PKT_CTRL_EMAC_IF_EGRESS_CLONE
- The data structure paPortMirrorConfig_t, paPktCaptureConfig_t and paEmacPortConfig_t are added to support interface-based port mirror and packet capture configuration. Refer to PA LLD doxygen document or pa.h for details.

Application code should follow following procedure to configure and enable this feature.

- Populate the interface-based configuration data structures and then call Pa_control() API with control code pa_CONTROL_EMAC_PORT_CONFIG and emacPort subtype pa_EMAC_PORT_CFG_MIRROR or pa_EMAC_PORT_CFG_PKT_CAPTURE to format PASS configuration packet and then send configuration packet to specified PASS input queue.
- Populate the packet control data structure (paPacketControlConfig_t or paPacketControl2Config_t) and system configuration data structure (paSysConfig_t) and then invoke Pa_control() API with control code pa_CONTROL_SYS_CONFIG to format PASS configuration packet and then send configuration packet to the specified PASS input queue.

To capture ingress packets at other processing stage such as post-IPSEC processing stage, application should setup a Rx command set with pa_CMD_MULTI_ROUTE command and then link that command set to the routing information (paRouteInfo_t or paRouteInfo2_t) of the desired final classification entry.

Please refer to pa/test/PAPktCapTest for some examples to configure and verify EMAC interface-based port mirroring and packet capture operations.

Please note that this feature is used for debug purpose only and it will reduce PASS throughput as much as 50% since each packet needs to be processed by PASS twice. Please note the egress packet capture or mirroring will be applied to pre-fragmented packets only due to hardware limitation.

- Ingress Default Route
  This feature allows host to configure PASS to send all packets with broadcast bit set (bit 2 of 1st mac header byte) from ingress port X to a corresponding route before or after the LUT look up. The ingress default route provides the route configurations for ingress broadcast (BC) and multicast (MC) packets and the unicast packets that do not match L2 entries on ingress port X as described below.

- Route BC/MC traffics prior to LUT1 lookup if configured as pre-classification route
- Route unmatched BC/MC traffics from EMAC port X if configured as post-classification route
- Route unmatched unicast traffic (post-classification) from EMAC port X if configured
- This rule precedes the exception route rule.

To minimize the performance impact when this feature is not enabled, there is a system-level global control in addition to the per interface configuration as described below:

- Following control bit definition is available at validBitMap and ctrlBitMap within paPacketControl2Config_t to enable/disable the system-wide ingress default route operation.
  - pa_PKT_CTRL_EMAC_IF_INGRESS_DEFAULT_ROUTE

- Additional interface paDefRouteConfig_t supports ingress default route configuration. Refer to PA LLD doxygen document or pa.h for details.

Application code should follow following procedure to configure and enable this feature.

- Populate the ingress default route configuration data structures and then call Pa_control() API with control code pa_CONTROL_EMAC_PORT_CONFIG and emacPort subtype pa_EMAC_PORT_CFG_DEFAULT_ROUTE to format PASS configuration packet and then send configuration packet to the specified PASS input queue.
- Populate the packet control data structure (paPacketControlConfig_t or paPacketControl2Config_t) and system configuration data structure (paSysConfig_t) and then invoke Pa_control() API with control code pa_CONTROL_SYS_CONFIG to format PASS configuration packet and then send configuration packet to the specified PASS input queue.

Please refer to pa/test/PAUnitTest/src/tests/test2.c for some examples to configure and verify Ingress Default Route operation.

- **Enhanced QoS Mode Operation**

  Enhanced QoS mode is an advanced priority-based routing algorithm where VLAN P-bit, IP DSCP or the EMAC port-based default priority is used to determine the destination QoS queue and CPPI flow. This routing algorithm is required to support egress L2 shaper and is described below.

  For each EMAC interface, PASS will be configured for:

  - Base queue (egress only)
  - Base flow (egress only
  - DSCP_MAP[] {one entry (= queue offset /flow offset) for each DSCP value, 64 total}

- VLAN_PRIORITY_MAP[] { one entry (= queue offset/flow offset) for each P-bit value, 8 total}
- Default priority to use on the interface (ingress only)
- Routing mode: DP-bit or DSCP
- PriorityOverride: True/False

Routing algorithm is:

- DP-bit mode
  - If frame has VLAN tag, use p-bits to lookup shaper input queue offset and flow from the VLAN_PRIORITY_MAP[] for the frame's egress port
  - If frame is un-tagged, but is an IP packet, use the DSCP value to lookup the shaper input queue offset and flow offset from the DSCP_MAP[] for the frame's egress port unless priority override is set for the egress port (see last bullet below).
  - If frame is un-tagged, and non-ip, then use the default priority for the frame's *ingress* port (or from host default global config for egress packets) to look up the shaper input queue offset and flow offset from the egress port's VLAN_PRIORITY_MAP[]. For from host/SOC-generated traffic, the default priority is a global configuration item
  - If priority override is set and the packet is IP then do as in un-tagged/non-ip (above bullet).
- DSCP mode
  - IP packets: use the packet DSCP bits and the DSCP_MAP [] for the egress port as above to determine the L2 shaper queue offset and flow offset to use
  - Non-ip packets: use the default priority for the frame's *ingress* port (or from-host default priority if packet is from host) to look up the shaper input queue offset and flow offset from the egress port's VLAN_PRIORITY_MAP[]. For SOC-generated traffic, the default priority is a separate configuration item.
  - Priority override setting is not applicable in this mode

  In modes, base queue and flow number is provided by host routing information for ingress traffic and the enhanced QoS mode per interface configuration parameters for egress traffic.

Enhanced QoS mode is triggered by host route information per ingress connection and it is enabled or disabled system-wide for all egress traffic destined to specified EMAC port. The related configuration data structures are described below:

- Following control bit definition is available validBitMap and ctrlBitMap within paPacketControl2Config_t to enable/disable the system-wide egress enhanced QoS route operation.
  - pa_PKT_CTRL_EMAC_IF_EGRESS_EQoS_MODE
- The new priority type pa_ROUTE_EQoS_MODE is added to paPriIntfRouteMode to indicate enhanced QoS mode for Host-route traffic.

- The data structure paEQosModeConfig_t is added to support enhanced QoS route configuration. Refer to PA LLD doxygen document or pa.h for details.

Application code should follow following procedure to configure and enable this feature.

- Populate enhanced QoS route configuration interface and then call Pa_control() API with control code pa_CONTROL_EMAC_PORT_CONFIG and emacPort subtype pa_EMAC_PORT_CFG_EQOS_MODE to format PASS configuration packet and then send configuration packet to the specified PASS input queue.
- Populate packet control data structure (paPacketControlConfig_t or paPacketControl2Config_t) and system configuration data structure (paSysConfig_t) and then invoke Pa_control() API with control code pa_CONTROL_SYS_CONFIG to format PASS configuration packet and then send configuration packet to the specified PASS input queue.
- For ingress connection which requires enhanced QoS route, invoke the LUT entry API such as Pa_addMac2() with priorityType set to pa_ROUTE_EQoS_MODE to format LUT configuration packet and then send this packet to the specified PASS input queue.

Please refer to pa/test/PAUnitTest/src/tests/test2.c and test4.c for some examples to configure and verify Enhanced QoS Mode operation.

This release includes resolved IPs as listed below:

IR Parent/ Child Number	Severity Level	IR Description
108550	Major	Support Packet Capture (Port Mirroring) feature in PA LLD (K2HK only)
108551	Major	Support Cascaded dual shaper in PA LLD (K2HK only)
112339	Major	PASS: Egress Flow exception causes internal PASS buffer leak
112340	Major	PASS: Deletion of the pair of IP/IPSEC entry may miss the IPSEC (SPI) entry due to firmware timing issue
112994	Major	PASS: Egress flow route to CPSW (pa_DEST_EMAC) does not work

Release 3.0.0.10

This release includes resolved IPs as listed below

IR Parent/ Child Number	Severity Level	IR Description
108066	Major	add RM to pa unit test on dsp

IR Parent/ Child Number	Severity Level	IR Description
108212	Minor	PA unit tests and examples should have the "All tests have passed" message when tests have passed to assure the proper automation result
108364	Major	Usermode LLD paUnit Test fails on k2hk platform
108377	Minor	PA LLD: IPv4 reassembly test failed due to stats mismatch in software-reassembly mode
112182	Major	PA LLD: Pa_delHandle() does not remove the root IP entry associated with an IPSEC entry

Release 3.0.0.9

To facilitate the integration effort of the Network Sub-System (NSS), which includes PASS, SASS and CPSW, the following two header files and the device-specific NSS layout configuration file nss_device.c are included at the PA LLD delivery package:

- nss_if.h: Define NSS transport layer related constants, such as number of CPPI Tx/Rx channels, the NSS Tx queue layouts and etc.
- nss_cfg.h: Define NSS transport layer and other global configuration data structure.
- device/k2x/nss_device.c: NSS device-specific configuration files
- added QMSS osal functions for Accumulator programming in test and examples

Although the NSS related files are included at PA LLD delivery package, they are not used by PA LLD itself. Instead, they are used by the NSS driver or application module such as NWAL and PA/SA[1] unit tests and examples to provide device-specific and/or device-independent libraries. All the PA/SA unit tests and examples are enhanced to demonstrate the usage of nss_if.h, nss_cfg.h and nss_device.c as described below:

- PA Unit Test: nss_if.h
- PA EMAC Example: nss_if.h, nss_cfg.h and nss_device.c
- PA Multicore Example: nss_if.h
- SA Unit Test: nss_if.h, nss_cfg.h and nss_device.c
- SA Basic Example:nss_if.h
- SA Multicore Example: nss_if.h

This release include all the bug fixes from PA LLD 2.0.1.5
This release also includes resolved IPs as listed below.

IR Parent/ Child Number	Severity Level	IR Description
107355	Major	Use DSP compiler options for debug ability with optimization (and compiler upgrade for the same)
107310	Major	PA LLD: Provide NSS Transport Layer related definitions and/or configuration data structure

[1] The SA example/unit test are under ti/drv/sa.

IR Parent/ Child Number	Severity Level	IR Description
103812	Major	PA LLD 3.0.0.x: Turn on C66x optimizer after C66x compiler version 7.4.2 bug ix fixed
106914	Major	Pa_delHandle() fails with error pa_INSUFFICIENT_CMD_BUFFER_SIZE
107045	Major	PaUnit Test user mode LLD fails on K2L
107090	Major	PA LLD: PASS generate incorrect IPv4 checksum for large size packets
107300	Major	PASS may deliver packets with unexpected CPPI flow to host queue
107412	Major	PA: Packets are lost when inner IP(IPSec) and post processing is enabled

Release 3.0.0.8

This release includes the following features from PA LLD 2.0.1.4:

- Interface-based routing support
- Enhanced user-defined statistics support

This release supports all keystone2 devices with two separate libraries for the first generation and second generation PASS respectively.

- Kepler/Hawking: PA
- Lamarr/Edison: PA2

The corresponding PASS firmware image files are moved to the following sub-directories respectively.

- Kepler/Hawking: fw/v0
- Lamarr/Edison: fw/v1

The top layer header file pa.h and pasahost.h are shared by both generations of PASS where the compiler switch NSS_GEN2 is used to distinguish the constant and macro definitions which are unique or different for the first and generation PASS. Besides, two addition sets of those constants and macros with suffix GEN1 and GEN2 for the first generation and second generation PASS respectively are provided to allow the PASS users to create single-library applications.

The latest PA LLD APIs are 99% backward compatible to support both generations of PASS to minimize the migration impact from the first generation PASS device to the second generation one. However, the application still needs to account for the transport layer changes summarized at the following table.

	PASS	PASS2
Number of CPPI Tx channels	9	21
Number of CPPI Rx channels	24	91
Number of CPPI Flows	32	96
Number of Tx Queues	9	21
L2 (MAC/SRIO) Configuration Queue	640	904
Outer IP Configuration Queue	641	905
Inner IP Configuration Queue	642	908
L4 (LUT2) Configuration Queue	643	908
Post-Classification Processing Queue	644	909
Tx Command Processing Queue	645	910

SASS Queue1	646	914
SASS Queue2	647	915
EMAC Queues	648	896-903
Outer ACL Configuration Queue	NA	904
Inner ACL Configuration Queue	NA	907
Outer IPSEC Configuration Queue	NA	905
Inner IPSEC Configuration Queue	NA	906
Egress Processing Stage 1 Queue	NA	910
Egress Processing Stage 2 Queue	NA	911
Egress Processing Stage 3 Queue	NA	912

Release 3.0.0.7

This release includes the following features from PA LLD 2.0.1.3:

- Multi-process support
- Linux user-mode support
- Enhance all examples and unit tests to be restartable

Release 3.0.0.6

This is the first official engineering drop of PA LLD for the second generation PASS (Packet Accelerator Sub-System) on advanced Keystone2 devices. The supported feature list is compatible with PA LLD version 1.3.0.11. To minimize the software migration effort of existing applications from older devices, the new PA LLD maintains backward compatibility of all existing APIs with the following minor exceptions:

- API Pa_requestStats() and Pa_formatStatsReply() are depreciated due to the new mechanism of handling PASS system statistics for the second generation PASS. It is no longer required to send the statistics request packet to PASS and wait for the statistics response packet from PASS. The application should call the new API Pa_querySysStats() which will return the formatted system statistics.
- The prototype of API Pa_configCrcEngine() stays the same. However, this API will configure the corresponding CRC engine directly in stead of formatting a CRC configuration command packet to be delivered to PASS.
- The following two parameters have been moved from CRC configuration structure paCrcConfig_t to CRC command structure paCmdCrcOp_t due to PASS CRC engine changes:
    - crc size
    - init value
- To add an IP entry of specific IP addresses and IPSEC SPI number with API Pa_addIp or Pa_addIp2, the application needs to specify the IP protocol as either IPSEC ESP (50) or IPSEC AH (51). The parameter proto should not be specified for an IP entry of specific SPI number only.

The second generation PASS provides several new features. Overview of new features and APIs are highlighted below, please refer to PA LLD header file pa.h, pa_fc.h or PA LLD doxygen document for details. The PA unit test program under pa/test provides some examples and sample codes for all new features.

- Outer IP and inner IP reassembly

  The second generation PASS provides a Reassembly engine (RA) which can be used to perform outer and inner IP reassembly and the reassembled packets will be delivered back to PASS for continuous processing. The following data structures and API are provided for RA related operation.
    - paRaConfig_t: RA global configuration structure added to PASS system configuration structure paConfig_t.
    - paRaGroupConfig_t: RA group configuration structure which is used by API Pa_control to control RA of Outer IP and inner IP reassembly operation.
    - paRaStats_t: Define IP reassembly statistics
    - Pa_queryRaStats: Query the RA statistics

- Egress Flow and Flow Cache operation

The second generation PASS includes one 256-entry LUT1 engine and three clusters of PDSP engine chain to support ingress packet forwarding, flow cache lookup and egress packet formatting and modification operation as described below:

  - Flow Cache Classification: Classify up to 256 established egress flows based on the inner IP and L4 parameters.
  - Egress Flow Operation: Perform up to 4-level packet modification such as IP mangling, IPSEC framing and encryption, L2 framing and etc per packet modification records.
  - Ingress Packet Forwarding: Route the ingress packets to the Egress processing unit as one of the classification routing options.

The following data structures and APIs are used for Egress Flow and Flow Cache related operations:

  - paEfRec_t: Define Egress Flow modification records
  - paFcInfo_t: Specify Flow Cache matching parameters
  - paEfOpInfo_t: Specify Egress Flow operation information
  - paFcStats_t: Define Flow Cache entry statistics
  - Pa_addFc: Add/Replace Flow Cache entry into Flow Cache lookup table
  - Pa_delFcHandle: Delete the specified Flow Cache entry
  - Pa_queryFcStats: Query Flow Cache per-entry statistics
  - Pa_configEflowRecords: Configure multiple Egress Flow modification records
  - Pa_configEflowExceptionRoute: Configure egress packet routing based on exception condition

- Pre-IPSEC and Post-IPSEC ACL (Access Control List) operation

The second generation PASS includes two 256-entry LUT1 with associated PDSP engines to support pre-IPSEC and post-IPSEC L3/L4 ACL lookup operation. The following data structures and APIs are used for ACL related operation.

  - paAclInfo_t: Specify ACL matching parameters
  - paAclConfig_t: Specify ACL actions per match
  - paAclStats_t: Define ACL entry statistics
  - Pa_addAcl: Add ACL entry into ACL table
  - Pa_delAclHandle: Delete the specified ACL entry
  - Pa_queryAclStats: Query the ACL per-entry statistics

- Local PKTDMA

The second generation NetCP includes a local PKTDMA (CPPI) and QMSS sub-system which may be used to transfer packets within the NetCP sub-system, such as packets from PA to RA, PA to SA and SA to PA. To use NetCP local PKTDMA, the application needs to enable this feature through the CPPI and QMSS LLDs and setup local PKTDMA path as the followings:

  - PA to RA: set control bit pa_RA_CTRL_USE_LOCAL_DMA at RA group configuration structure paRaGroupConfig_t.
  - PA to SA: set routing destination to pa_DEST_SASS_LOC_DMA in stead of pa_DEST_SASS
  - SA to PA: set control bit sa_DEST_INFO_CTRL_USE_LOC_DMA at the SA LLD channel destination configuration structure Sa_DestInfo_t.

The following features, which are still supported functionally or in API by PA LLD for backward compatibility, should be depreciated or changed due to the more advanced capability of the second generation PASS:

- **PASS-assisted IP Reassembly**: This feature is still supported, but it is recommended to be replaced by the RA-based IP reassembly operation which does not request host intervention.
- **IPSEC ESP NAT-T Detection:** In the 2nd generation PASS, the IPSEC NAT-T detector is implemented within the Outer IP and IPSEC processing stage to avoid the re-entry operation from LUT2 stage. And the detector is also implemented at the traditional LUT2 stage to maintain backward compatibility. It is recommend making the following two changes to enable IPSEC NAT-T detector at the appropriate processing stage to avoid PASS throughput degradation due to IPSEC NAT-T packet re-entry operation.

    - IPSEC NAT-T configuration: set control bit pa_IPSEC_NAT_T_CTRL_LOC_LUT1
    - Outer IP routing configuration: use pa_DEST_CONTINUE_PARSE_LUT1 in stead of pa_DEST_CONTINUE_PARSE_LUT2
- **Tx Commands:** Most of tx command operations such as IP/UDP checksum, IPSEC AH patching and IP Fragmentation are also supported by the Egress Flow operation. To simplify the egress operation, it is recommended to setup an egress flow and replace a set of Tx commands with a single pa_CMD_EF_OP command.
- **GTPU classification with L3 link:** This feature is no longer required since the advanced LUT2 engine supports GTPU 32-bit Tunnel-ID classification with L3 link and therefore, it is not necessary to restrict the effective tunnel-ID to 24-bit. The GTPU configuration will be still processed by the PASS for backward compatibility and the configuration command packet will be ignored by PASS.
- **Configurable L3 offset location:** This feature is no longer required since both offsets to outer IP and inner IP will be provided at packet information area and extracted by the following two Macros:
    - PASAHO_LINFO_READ_L3_OFFSET()**:** Offset to outer IP
    - PASAHO_LINFO_READ_INNER_IP_OFFSET(): Offset to inner IP

Although most of the PA LLD APIs are backward compatible, some minor application changes are still required due to the following transport layer changes introduced at the second generation keystone2 devices:

- More PASS CPPI channels: There are 21 Tx channels and 91 Rx channels in the second generation NetCP devices where there are 9 Tx channels and 24 Rx channels in the first generation devices.
- More PASS Tx Queues: The are 21 Tx queues in the second generation NetCP devices where there is only 9 TX queues in the first generation devices.
- Local PKTDMA and QMSS: There is NetCP local PDTDMA and QMSS modules in addition to the global ones.
- NetCP Tx Queue Layout:

Location	NetCP 1.0 equivalent	Global	Local
EMAC priority 0	EMAC	896	0
EMAC priority 1	NA	897	1
EMAC priority 2	NA	898	2
EMAC priority 3	NA	899	3
EMAC priority 4	NA	900	4
EMAC priority 5	NA	901	5
EMAC priority 6	NA	902	6
EMAC priority 7	NA	903	7
Ingress 0	PDSP0: MAC/SRIO	904	8
Ingress 1	PDSP1: Outer IP	905	9
Ingress 2	NA	906	10
Ingress 3	NA	907	11
Ingress 4	PDSP2 (Inner IP) and PDSP3 (LUT2)	908	12
Post	PDSP4: Command Set	909	13
Egress 0	PDSP5: Tx Command	910	14
Egress 1	NA	911	15
Egress 2	NA	912	16
RA (Reassembly Engine)	NA	913	17
SASS	SASS	914	18
SASS2	SASS2	915	19
Statistics Module	NA	916	20

Release 3.0.0.1 – 3.0.0.5:

TI internal engineering drops

Release 3.0.0.0:

This is the initial engineering drop of the second generation PASS (Packet Accelerator Sub-System) on advanced Keystone2 devices. The supported feature list is compatible with PA LLD version 1.2.3.3. It is provided for initial integration support for the AVV team. Only limited tests have been performed at the Lamarr simulator with local provided tisim_pass.dll.

Release 1.3.0.11

Resolved IRs as listed below

IR Parent/ Child Number	Severity Level	IR Description
00101118	Major	Received Fragmented ICMP with invalid fragment-offset causes eth0 to jam

Release 1.3.0.10

Resolved IRs as listed below:

IR Parent/ Child Number	Severity Level	IR Description
00100400	Minor	PA LLD: 1.3.0 software manifest points to release 1.2.0
00100401	Minor	PA LLD: Need constant definition of the virtual link buffer ID

Release 1.3.0.9

- Resolved IRs as listed below

IR Parent/ Child Number	Severity Level	IR Description
00099556	Major	PA LLD: Deleting LUT2 entries enables GTPU processing unconditionally

Release 1.3.0.8

- API changes involved in this release are backward compatible to PA 1.3.0.7; no application modification is required for PA 1.3.0.7 based features. Details of new feature and API changes from PA LLD 1.3.0.7 are described below:

    - **GTPU classification with L3 link**

    Due to the LUT2 engine using 32-bit matching parameter, the default GTP-U classification is solely based on its 32-bit tunnel ID. However, it may be desirable to match a GTP-U tunnel with both its tunnel ID and the previous link information at some use cases. PASS is enhanced to support GTPU classification with L3 link where the classification vector consists of the least significant 24-bit of tunnel ID and an 8-bit previous link parameter.

    The data structure paCtrlInfo_t is enhanced to include global GTPU configuration parameter gtpuCfg. The application should invoke API Pa_control() with configuration code set to pa_CONTROL_GTPU_CONFIG to enable/disable this feature at system startup.

    - **Configurable L3 offset location**

    The PASS records several protocol header offsets as part of packet information stored at PS Info section while it is parsing the packet. In the current implementation, the L3 offset will point to the outer IP header. However, it is useful to set L3 offset to the packet offset of the inner IP header at certain use cases.

    To support configurable L3 offset at the packet info, a new packet control bit pa_PKT_CTRL_L3OFFSET_TO_INNER_IP is defined, which can be used to enable/disable this feature at system startup.

    - **Cascaded Forwarding**

    The cascaded forwarding packets are expected to be delivered to QoS queues based on the VLAN/DSCP priority at its final classification stage. Therefore, those packets should not be subject to some PASS actions such IP reassembly and IP fragment exception route.

    A new packet routing destination type pa_DEST_CASCADED_FORWARDING_LUT1 is added, which should be used at Pa_addMac() API call to disable IP reassembly and IP fragmentation exception route at sub-sequent classification stage.

    - **Priority-based routing with post-classification command set**

    There is some use cases where output packets from QoS are delivered to PASS for pre-routing operation such as tx timestamp report and both egress and ingress forwarding packets go through the same QoS. To support this use case, PASS is enhanced to delay the post-classification command set execution until the packets re-entering PASS from QoS if priority-based routing is selected..

- Resolved IRs as listed below:

IR Parent/ Child Number	Severity Level	IR Description
00098539	Major	PA LLD: Support GTPU classification with L3 link
00098540	Minor	PA LLD: Support configurable L3 offset location

IR Parent/ Child Number	Severity Level	IR Description
00098877	Minor	PA LLD: Enhance IPv6 fragmentation and reassembly to support IPv6 extension headers
00098878	Major	PA LLD: Support Cascaded Forwarding
00099016	Major	PA LLD: Need to support multiple UDP 2152 entries
00098922	Minor	paEthInfo2_t structure defines validBitMap, but there are no constants defined to enable setting the bitmap
00098920	Major	Pa_addMac2 () not implemented

Release 1.3.0.7

- Resolved IRs as listed below

IR Parent/ Child Number	Severity Level	IR Description
00098035	Major	PA LLD: Unexpected Cache operations related to Virtual Link cause potential memory corruption

Release 1.3.0.6

- This release includes Custom LUT2 operation enhancement by adding a new parameter custHdrSize to API Pa_setCustomLUT2. The custHdrSize specifies size of fixed-length custom header; it is used by PASS to adjust the location of the next protocol header so that the packet can be further processed by another module such as SASS or host application. The custHdrSize should be set to 0 for variable-length header.
- Resolved IRs as listed below

IR Parent/ Child Number	Severity Level	IR Description
00097383	Minor	PA LLD 48-bit Time Stamp macro from host file is inconsistent with firmware definition
00097730	Major	PA LLD: Custom LUT2 enhancement

Release 1.3.0.5:

- Resolved IRs as listed below

IR Parent/ Child Number	Severity Level	IR Description
00097274	High	PA_emacExample_exampleProject fails on EVM and Simulator for BE and LE
00097275	High	PA_multicoreExample_exampleProject fails on EVM and Simulator for BE and LE

Release 1.3.0.4:

- Resolved IRs as listed below

IR Parent/ Child Number	Severity Level	IR Description
00097286	High	Added support for additional comamnds for IP fragmentation

Release 1.3.0.3:

API changes involved in this release are backward compatible to PA 1.3.0.2, no application modification is required for PA 1.3.0.2 based features, except PA initialization regarding virtual link. In order to use the Virtual Link feature, one must initialize PA with the maximum number of desired virtual links by setting paSizeInfo_t->nVlnkMax to a non-zero value. Consequently, if virtual link feature is not used, it must be turned off by setting paSizeInfo_t->nVlnkMax to 0. Release includes additional feature enhancements. Details of new feature and API changes from PA LLD 1.3.0.2 is described below:

- **Virtual Link**
  This feature was added to support linkage sustainability between Outer IP and Inner IP LUT entries during IPSec Tunnel rekey. Previous APIs was not changed to maintain backwards compatiblity, while the new API functions Pa_addIP2(), Pa_addVirtualLink() and Pa_delVirtualLink() provides support for Virtual Link.

  Pa_addIP2() requires the following parameters, changes from Pa_addIP() are highlighted
  - `Pa_Handle iHandle,`
  - `paIpInfo2_t *ipInfo,`
  - `paParamDesc *params,`
  - `paLnkHandle_t *retHandle,`
  - `paCmd_t cmd,`
  - `uint16_t *cmdSize,`
  - `paCmdReply_t *reply,`
  - `int *cmdDest`

  The new parameters in Pa_addIP2() were designed for improving expandability for future API changes:
  - `paIpInfo2_t` is extended upon `paIpInfo_t` to include a valid bit map
  - `paParamDesc` combined several conditionally optional parameters with a valid bit map

  The following are the sequence of API calls in pseudo-code to correctly invoke Virtual Link feature:

- Tunnel configuration
  - addMac()
  - virtLink = Pa_addVirtualLink().
  - Outer IP Rule-> Call Pa_addIP2() with
    - paParamDesc-> nextLink = virtLink
    - paParamDesc->prevLink = NULL or L2 handle
  - Inner IP Rule -> Call Pa_addIP2() with
    - paParamDesc-> nextLink = NULL
    - paParamDesc->prevLink = virtLink

- **QoS based priority routing**
  This feature was added to facilitate QoS based priority routing using either VLAN or DSCP values from individual packets. The new routing logic enables matched packets to be delivered to different QoS queues derived by taking the base queue number added with either VLAN/DSCP. We provide a new structure paRouteInfo2_t, which extends upon the original paRouteInfo_t, where the parameter priorityType is used to specify packet routing priority mode. Valid values are:
  - pa_ROUTE_PRIORITY_NONE = 0 (Default)
  - pa_ROUTE_PRIORITY_VLAN, Route by using VLAN bits as priority
  - pa_ROUTE_PRIORITY_DSCP, Route by using DSCP bits as priority

- Resolved IRs as listed below

IR Parent/ Child Number	Severity Level	IR Description
00095408		Add IPv6 fragmentation and reassembly unit test

Release 1.3.0.2:

- Merged changes from release 1.2.3.2.
- Resolved IRs as listed below
- Provided as internal engineering drops

IR Parent/ Child Number	Severity Level	IR Description
00095730	Major	PA LLD: Buffer leak through PASS

Release 1.3.0.1:

- Added IPv6 reassembly assistance to forward fragment packets to software reassembly stack
- Added IPv6 fragmentation support in firmware
- Added support for 48-bit transmit and receive timestamp.
- Added the following two counters as of PASS system statistics
  - nIpv4PacketsInner: Number of Inner IPv4 packets
  - nIpv6PacketsInner: Number of Inner IPv6 packets
- Resolved IRs as listed below

IR Parent/ Child Number	Severity Level	IR Description
00093442	Major	Provide 48-bit timestamp in descriptor

Release 1.3.0.0:

- Provided as internal engineering drops

Release 1.2.3.3:

- Resolved IRs as listed below

IR Parent/ Child Number	Severity Level	IR Description
00095942	Minor	PA_UnitTest_testProject fails on the simulator

Release 1.2.3.2:

- Add new post-classification command pa_CMD_SPLIT which is used to split the packet into header and payload portion to be delivered to different destination queues with different CPPI flows.

  Please note that the first 8-byte of psInfo area is reserved for this splitting operation, therefore, they should not be updated by the pa_CMD_COPY_DATA_TO_PSINFO commands within the same command set.

- Resolved IRs as listed below

IR Parent/ Child Number	Severity Level	IR Description
00094669	Minor	Update of packet length for incoming RX packets without CRC bytes
00095226	Major	PA LLD: Add Packet splitting feature

Release 1.2.3.1:

- Resolved IRs as listed below

IR Parent/ Child Number	Severity Level	IR Description

IR Parent/ Child Number	Severity Level	IR Description
00094662	Minor	PA LLD: Pa_resetControl(Query) returns wrong state
00094975	Major	PA LLD: Invalid LUT1 entry may be added into LUT1 engine under race condition

Release 1.2.3.0:

- Resolved IRs as listed below:

IR Parent/ Child Number	Severity Level	IR Description
00093638	Major	UDP Lookup fails if the IPv6 packet contains extension headers
00094117	Minor	PA LLD: PA LLD Bit-manipulation macros cause compiler warning at Linux Kernel build

Release 1.2.2.2:

- Added new API Pa_getPDSPVersion() to query the version number of PDSP image to provide the capability to verify the compatibility of the PA LLD and the PDSP image which may be downloaded by another processor. The version number of the PDSP image should be identical to the version number of the PA LLD.
- Resolved IRs as listed below:

IR Parent/ Child Number	Severity Level	IR Description
00091769	Minor	PA LLD: Add API to query the version number of PDSP images
00092662	Major	PA LLD: a_CMD_PATCH(delete) causes extra bytes to be deleted at pa_CMD_REMOVE_TAIL operation
00092663	Major	PA LLD:Some of IPv4 fragments are not delivered to the desired destination queue
00092664	Major	PA LLD: Invalid IPv4 header causes the CPSW stop receiving packets

Release 1.2.2.1:

- Resolved IRs as listed below:

IR Parent/ Child Number	Severity Level	IR Description
00091702	Major	PA LLD: Enhanced Cache-related OSAL functions per Cache Advisory 12
00091767	Minor	PA LLD: Enhance IPv4 fragmentation operation to support fragment padding

Release 1.2.2.0:

Release includes additional feature enhancements as per PA 1.2.2 requirements. Details of new feature and API changes from PA LLD 1.2.1.2 is highlighted below:

- **System statistics enhancements**

The PASS system statistics has been revised by removing unused statistics and adding some useful statistics. The following system statistics are removed:

- Classify1.nNonIpPackets
- Classify1.nCommandFail
- Classify1.nInvalidComReplyDest
- Classify2.nParseFail
- Classify2.nInvldHdr
- Classify2.nCommandFail
- Classify2.nInvalidComReplyDest
- Common.nIdAllocationFail

The following system statistics are added:

- Classify1.nSrioPackets
- Classify1.nTxIpFrag
- Classify2.nPackets

- **Miscellaneous Packet Control enhancements**

Replace the global configuration data structure paPacketVerifyConfig_t with paPacketControlConfig_t to support general packet control in addition to the enhanced protocol header verification for the specified protocol. New parameters are added to specify the desired user-defined statistics of the rx padding error counter and the tx padding counter. The parameters protoBitMap is replaced with ctrlBitMap as defined below:

- pa_PKT_CTRL_HDR_VERIFY_PPPoE: Enable/Disable enhanced error check for PPPoE header
- pa_PKT_CTRL_HDR_VERIFY_IP: Enable/Disable enhanced error check for IP header
- pa_PKT_CTRL_MAC_PADDING_CHK: Enable/Disable MAC (802.3) padding error check
- pa_PKT_CTRL_IP_FRAGS_TO_EROUTE: Enable/Disable IP fragments routing through exception route

- The next route command pa_CMD_NEXT_ROUTE is enhanced to support L2 padding control in the to-network direction.
- Add command pa_CMD_VERIFY_PKT_ERROR to control the destination of the packet with the specified checksum or CRC error.
- Resolved IRs as listed below:

IR Parent/ Child Number	Severity Level	IR Description
00090761	Major	PA: Redirection of unsupported protocol through Next Fail route
0091012	Minor	PA LLD: Buffer leak found at all examples
00091105	Minor	PA LLD: Add Verify Checksum (Packet Error) Command
00091385	Major	PA LLD: L2 (MAC) Padding Support
00091386	Major	PA LLD: Provide IP Fragment counters
00091387	Major	PA LLD: Enhance IP Fragments routing options

Release 1.2.1.2:

Release includes additional feature enhancements as per PA 1.2.1 requirements. Details of new feature and API changes from PA LLD 1.2.1.1 is highlighted below:

- **User-defined statistics enhancements**

The user-defined statistics feature is enhanced to support up to 512 statistics consisting of some 64-bit counters and some 32-bit counters whereas the total size of all counters cannot exceed 2048 bytes**.** The parameter (num64bCounters) is added into the user-defined statistic configuration data structure (paUsrStatsConfig_t) to specify the number of 64-bit counters.

The user-defined statistics query API Pa_requestUsrStats() is enhanced to provide the user-defined statistics at return and the corresponding API Pa_formatUsrStatsReply() is removed.

- **IPSEC NAT-T packet detector**

The IPSEC NAT-T detector identifies the IPSec NAT-T packets with UDP source port number or UDP destination port number equal to the specified UDP port number such 500 or 4500.

  - Identify the IPSec NAT-T negotiation packet (with SPI field equal to zero) and forward it to the queue specified by exception route pa_EROUTE_NAT_T_CTRL

- Identify the IPSec NAT-T service packet (with SPI field does not equal to zero) and forward it to the queue specified by exception route pa_EROUTE_NAT_T_CTRL
- Identify the IPSec NAT-T keepalive packet and forward it to the queue specified by exception route pa_EROUTE_NAT_T_KEEPALIVE.

The IPSEC NAT-T detector is disabled as PASS startups. The module user can call API Pa_control() with the NAT-T configuration data structure paIpsecNatTConfig_t to enable and configure this detector at PASS.

- **802.1ag packet detector**

The 802.1ag detector identifies the 802.1ag packet in ether draft or standard format per module user configuration. The PASS delivers all 802.1ag packets to the queue specified by exception route pa_EROUTE_802_1ag.

The 802.1ag detector is disabled by default since the 802.1ag packets may be detected and routed by a general MAC rule with etherType set to 0x8902. The module user should call API Pa_control() with the 802.1ag configuration data structure pa802p1agDetConfig_t to enable and configure this detector at PASS.

- **Protocol Header verification enhancements**

The new global configuration data structure paPacketVerifyConfig_t is used to enable/disable the enhanced protocol header verification for the specified protocol. PASS supports the following two enhanced protocol verification. The error packet route is specified by the corresponding exception route such as pa_EROUTE_PPPoE_FAIL and pa_EROUTE_IP_FAIL:

- PPPoE header verification:
  - Version = 1
  - Type = 1
  - Code = 0
- IPv4 header verification:
  - Header length >= 20
  - Total length > 20
  - Source address is not broadcast
  - Destination address is not 0
  - TTL is not 0

Please note that the enhanced protocol header verification will reduce the packet throughput. Therefore, it is recommended to be disabled.

- **Protocol Indication within the packet information**

The following MACROs are provided at pasahost.h to verify whether the received packets contain the specific protocol header:

- PASAHO_LINFO_IS_MAC(x)
- PASAHO_LINFO_IS_802_3(x)
- PASAHO_LINFO_IS_WITH_VLAN(x)

    - PASAHO_LINFO_IS_WITH_MPLS(x)
    - PASAHO_LINFO_IS_PPPoE(x)
    - PASAHO_LINFO_IS_IP(x)
    - PASAHO_LINFO_IS_IPSEC_ESP(x)
    - PASAHO_LINFO_IS_IPSEC_AH(x)
    - PASAHO_LINFO_IS_UDP(x)
    - PASAHO_LINFO_IS_UDP_LITE(x)
    - PASAHO_LINFO_IS_TCP(x)
    - PASAHO_LINFO_IS_GRE(x)
    - PASAHO_LINFO_IS_GTPU(x)
    - PASAHO_LINFO_IS_CUSTOM(x)
    - PASAHO_LINFO_IS_SCTP(x)
    - PASAHO_LINFO_IS_IPSEC_NAT_T(x)
- The patch command pa_CMD_PATCH_DATA is enhanced to support deletion operation by adding the control bit definition pa_PATCH_OP_DELETE.
- Update the multi-route entry data structure to support optional swInfo0 update per entry.
    - Add control bit definition pa_MULTI_ROUTE_REPLACE_SWINFO
    - Add parameter swInfo0
- Resolved IRs as list below

IR Parent/ Child Number	Severity Level	IR Description
00088844	Major	Enhance User Defined Statistics feature to support up to 512 32-bit counters
00089460	Major	Additional handling of alignment requirement in PA helper function Pa_formatTxCmd() for creating Tx commands
00089901	Major	pa_addMac() and pa_addIp() functions is not returning handle during valid duplicate mac entry
00089978	Major	PA LLD: Support IPSEC NAT-T detection
00089980	Major	PA LLD: Support 802.1ag Packet Detection
00089993	Major	PA LLD: Enhance PPPoE and IP Protocol Header Error Processing
00089997	Major	PA LLD: Add more protocol indication bits in the Packet Information
00090132	Major	PA LLD: Enhance multiroute operation to support optional swInfo0 update per entry
00090134	Major	PA LLD: Enhance PA command pa_CMD_PATCH_DATA to support deletion

Release 1.2.1.1:

- Added support for Resource Manager LLD. For all existing applications there are no API modifications required. The Pa_startCfg API has been added to configure use of the RM LLD if desired.

Release 1.2.1.0:

- LUT1 configuration APIs are enhanced to allow application to specify the desired LUT1 instance. This enhancement allows LUT1 re-entry to support some advanced IP layer operation such as IPSEC ESP over IPSEC AH. The parameter " lutInst" is added to the following APIs
    - Pa_addIp
    - Pa_addCustomLUT1

  To maintain backward compatibility, set " lutInst" to

  pa_LUT_INST_NOT_SPECIFIED.
- The message length patching command pa_CMD_PATCH_MSG_LEN is added to instruct the PASS to update the message length field within some L2 protocol header such as 802.3 and PPPoE after the potential IP fragmentation operation.
- Add new API Pa_getTimestamp() to query the 48-bit PASS system timestamp.
- Add the following MACROs to extract egress packet information:
    - PASAHO_LINFO_READ_MAC_PKTTYPE():Extract the MAC packet type(Unicast, Multicast or Broadcast)
    - PASAHO_LINFO_READ_INPORT():Extract the (1-based) input EMAC (SGMII) port number
- Redefine PA LLD OSAL functions to be consistent with the ones used at other LLDs:
    - Cache coherency protection: Pa_beginMemAccess() & Pa_endMemAccess()
    - Multi-core or Multi-thread access protection: Pa_osalMtCsEnter() & Pa_osalMtCsExit()
- Resolved IRs as listed below

IR Parent/ Child Number	Severity Level	IR Description
00086799	Major	Enhance LUT1 configuration API to support IPSEC ESP over AH operation
00086801	Minor	Need to handle EMAC psFlags when routing destination is set to Host

IR Parent/ Child Number	Severity Level	IR Description
00087268	Major	Record and report input SGMII port number
00087269	Major	Enhance multi-route operation to allow one of the multi-route entry forwarding the packet back to PASS for “continue parsing"
00087270	Major	Support IP over PPPoE
00087778	Major	PA LLD: Add API to read the current PASS timestamp
00087779	Major	PA LLD: Define new OSAL function for multi-core protection
00087922	Major	MAC packet type information required for incoming packet from PA
00088269	Minor	Increasing the number of custom LUT2 types to enable core to core communication
00088842	Major	CRC + Patch Command enhancement

Release 1.2.0.3:

- Release adds examples and unit test code to demonstrate Linux User Mode LLD usage for ARM processor. Support only applicable for devices with ARM processor.

Release 1.2.0.2:

- Release includes modifications to support User Mode access for ARM processor. Support only applicable for devices with ARM processor.

Release 1.2.0.1:

- Fix for IR: 00086230: PASS drops SCTP packets

Release 1.2.0.0:

Release includes additional enhancements as per PA 1.2 requirements. Details of new feature and API changes from PA LLD 1.1.1.0 is highlighted below:

- **User-defined statistics**

  The PA LLD and PASS maintain up to 256 user-defined hierarchical statistics which consists of 64 64-bit counters and 192 32-bit counters. Each statistic is classified to a level which can be linked to one of the next level statistics. When one counter is incremented, all counters in its linking chain will be incremented, too. The following

APIs and data structures are added or enhanced to support this and other new features:

- The parameter nUsrStats, which defines the maximum number of user-defined statistics, is added to the data structure paSizeInfo_t.

  The PA LLD will require a new buffer of the user-defined statistics link table if nUsrStats is set to non-zero value.

  To maintain backward compatibility, set nUsrStats to zero.

- Call the new API Pa_control() with the user-defined statistics configuration structure paUsrDefinedStatsConfig_t to update the number of user statistics dynamically.
- Call the new API Pa_configUsrStats() to configure a list of user-defined statistics for their type (byte counter or packet counter) and link (to the next layer counter).
- Specify the index of the user-defined statistics of the new command structure paCmdUsrStats_t or paCmdSetUsrStats_t which will be used as part of the routing information (paRouteInfo_t).
- Call the new API Pa_requestUsrStats() and Pa_formatUsrStatsReply() to inquire and format the user-defined statistics respectively.

- **IPv4 Fragmentation**

The IP fragment command pa_CMD_IP_FRAGMENT is added to instruct the PASS to perform IPv4 fragmentation operation. Packets are sent to PASS PDSP5 with both IP fragment command and next route command which specifies the final destination, the entire packet or its fragments will be delivered to the final destination based on the packet size and the MTU size specified at the IP fragment command. This operation can be applied to non-IPSEC packets, inner IP prior to IPSEC encapsulation and outer IP after IPSEC encapsulation.

For the inner IP fragmentation, follow the following procedure:

1. Host sends packets with the IP fragment command and the next route destination queue set to a host queue to PASS PDSP5 for IP fragmentation operation.
2. All fragments will be delivered to the specified host queue.
3. Host adds the outer MAC/IP header, invokes the SA LLD API sendData () and then sends the fragments to the SA queue.
4. Each fragment will be encrypted, authenticated and forwarded to the final destination.

For the outer IP fragmentation, the overall operation is stated below:

1. Packet is delivered to SASS for IPSEC operation
2. Packet is delivered to PASS for IP Fragmentation operation
3. The entire packet or its fragments are delivered to the network.

The next route command is required for step 2.

- **PASS-assisted IP Reassembly**

The current version of PASS does not support IP reassembly; all the IP fragments are detected, forwarded to and reassembled at host. The reassembled IP packet may be forwarded back to PASS for continuous classification. The drawback of this approach is that the order of the incoming packets is not guaranteed to be maintained.

To provide better support for IPv4 reassembly, the PA-assisted IP Reassembly operation is introduced and summarized below:

- Array of traffic flows which consist of source IP, destination IP, protocol and counter are maintained at PASS.
- Traffic flow is activated by the PASS when the first IP fragment is detected and forwarded.
- Traffic flow is freed when its packet count reaches 0
- All packets belong to any active traffic flow will be forwarded to the host so the packet order will be maintained.
- Number of active traffic flow is configurable [0, 32]
- IP fragments is forwarded to host with "none" traffic flow id if no traffic flow is available. In this case, the packet order is not guaranteed to be maintained.

The host IP reassembly module, which is not part of PA LLD, should interact with the PASS and perform the full IP reassembly operation. An IP Reassembly sample code, which demonstrates how to interact with the PASS to perform IPv4 reassembly, is available for reference at ti\drv\pa\example\reassemLib.

The PASS-assisted IP reassembly feature is disabled by default. To enable and configure this feature, call the new API Pa_control() with IP reassembly configuration structure paIpReassmConfig_t to prepare and send the command packet to the corresponding PDSP. The outer IP (PDSP1) and inner IP (PDSP2) can be configured independently.

- **Atomic queue diversion operation per LUT2 entry replacement**

This feature is provided to support handover operation. Enhance the API Pa_addPort() and Pa_addCustomLUT2() to support the atomic queue diversion operation, which means that the QMSS moves the entries in the diverted queue to the destination queue, if the divertQ is specified and fReplace flag is set. In this case, the PASS will complete the LUT2 update, wait for the queue diversion to be complete and then resume processing incoming packets. Following is the additional parameter at both APIs:

  - divertQ: specify the source queue for atomic queue diversion with LUT2 update

  To maintain backward compatibility, set

    - " divertQ" to pa_PARAMS_NOT_SPECIFIED

- **Post-classification command set enhancements**

The PASS will support either 64 of 64-byte command sets or 32 of 128-byte command sets. It support 64 command sets by default. To change the number of command sets, call the new API Pa_control() with the command set configuration structure paCmdSetConfig_t to format and send the configuration command packet to PASS.

- The EMAC classification data structure paEthInfo_t is enhanced to include the input EMAC port (inport) as an optional classification criterion.

  To maintain backward compatibility, set inport to 0.

- Enhance the blind patch command structure to support MAC header replacement operation: replace the single Boolean parameter "write" with "ctrlBitfield".
- Resolved IRs as listed below:

IR Parent/ Child Number	Severity Level	IR Description
00083432	Major	PA LLD: CRC operation enhancement by allowing lenOffset to be negative
00083010	Major	PA LLD: Classification based on ingress port number
00083375	Minor	PA LLD: Additional defines to avoid using hard coded values in PA interfaces

Release 1.1.1.0:

- Documentation (pa.h)
  - paCrcConfig_t: Add example for 16-bit CRC
  - paCmdNextRoute_t: Clarify how the next route command is used in the from-network direction.
- Resolved IRs

IR Parent/ Child Number	Severity Level	IR Description
00083715	Major	PA LLD: CRC does not work
00083377	Minor	Mismatch between the OSAL begin and end tags

Release 1.1.0.10:

- Update OSAL functions Osal_paBeginMemAccess() and Osal_paEndMemAccess() at examples and unit tests
- Resolved IRs

IR Parent/ Child Number	Severity Level	IR Description
00081280	Major	Packets not received when Udp port entry with destination port 2152 (GTPU port number)
00081493	Minor	Enhance PA Multicore example with cache enabled

Release 1.1.0.9:

- Cleanup OSAL functions at examples and unit tests
- Resolved IRs

IR Parent/ Child Number	Severity Level	IR Description
00080511	Major	Pa_resetControl() API may block forever when called while receiving incoming packets

Release 1.1.0.8:

- Support C66 ELF only
- Add multicore example
- Both the next route data structure paCmdNextRoute_t and the routing info data structure paRouteInfo_t are enhanced to support EMAC port control when the packet destination is set to pa_DEST_EMAC.

  To maintain backward compatibility, search and replace pktType with pktType_emacCtrl.
- The data structure of CRC operation command paCmdCrcOp_t is enhanced to include new parameter frameType.
- Resolved IRs

IR Parent/ Child Number	Severity Level	IR Description
00071334	Major	Add PA example for multicore routing of packets
00080279	Major	PA LLD: Enhance ETH routing to specify the destination EMAC port
00080280	Major	PA LLD: Enhance CRC operation to support variable payload length and offset for supported frame types such as WCDMA FP HS-DSCH Data Frame type 2 and type 3

Release 1.1.0.7:

- Resolved IRs

IR Parent/ Child Number	Severity Level	IR Description
00078735	Major	Duplicate MAC entry puts the route in pending state. Request to change it to Active while in transition
00078888	Major	IP Fragmentation Problem on NetCP

Release 1.1.0.6:

Release includes additional enhancements as per PA 1.1 requirements. Details of API changes from PA LLD 1.1.0.5 is highlighted below:

- LUT1 configuration APIs enhanced to support LUT1 index configuration from application. The parameter " index" has been added to the following APIs
    - Pa_addMac
    - Pa_addIp
    - Pa_addCustomLUT1.

  To maintain backward compatibility, set " index" to

  pa_LUT1_INDEX_NOT_SPECIFIED.
- Enhance the API Pa_addPort() to support both 16-bit UDP/TCP port and 32-bit GTPU Tunnel ID. Following are two additional parameters :
    - Port size: specify the LUT2 classification parameter size as 16-bit or 32-bit
    - Replace flag: indicate whether the LUT2 entry exist or not

  To maintain backward compatibility, set

    - " Port size" to pa_LUT2_PORT_SIZE_16
    - " Replace flag" to 0.
- Additional API Pa_addSrio() to support SRIO L0-L2 classification.

- Rename and enhance the following APIs to support the enhanced Custom LUT1 and LUT2 classification per PA 1.1 requirements.
    - Pa_setCustomL3 ➔ Pa_setCustomLUT1
    - Pa_addCustomL3 ➔ Pa_addCustomLUT1
    - Pa_setCustomL4 ➔ Pa_setCustomLUT2
    - Pa_addCustom ➔ Pa_addCustomLUT2
- API Pa_configRouteErrPacket() renamed to Pa_configExceptionRoute() and includes support for general exception routes including the error routing.
- Following new APIs are added for the multi-route, CRC engine and system timestamp configuration respectively:
    - Pa_configMultiRoute
    - Pa_configCrcEngine
    - Pa_configTimestamp
- Following new APIs are added for to-network and from-network post-match command processing:
    - Pa_formatTxCmd
    - Pa_configCmdSet
- The type of parameter cmdSize in the following two APIs is changed from int to uint16_t to be consistent with all other APIs.
    - Pa_formatTxRoute
    - Pa_formatRoutePatch
- PA system APIs Pa_resetControl and Pa_downloadImage are updated to include the Pa_handle as input parameter. With this enhancement, the PA LLD supports multiple PA instance.
- The parameter handle in the API Pa_delHandle is changed from paHandleL2L3_t o a pointer to the entry handle so that the entry handle can be set to NULL to indicate that it has been deleted.
- PA initialization configuration structure is enhanced to include the following two parameters:
    - initDefaultRoute: specify whether the default traffic flow should be initialized
    - baseAddr: specify the PASS base address which is defined at
      “ ti/csl/cslr_device.h”
- The data structure paHandle_t is renamed to paEntryHandle_t to clarify its usage and avoid the confusion with Pa_handle. The paEntryHandle contains either the l2l3Handle or the l4handle when the API call Pa_forwardResult returns in response to an LUT1/LUT2 entry insertion request. Please note that the original

structure paHandle_t contain the pointer to the l4handle in stead of the l4handle itself.

- IP Lookup Information data structure paIpInfo_t is enhanced to support SCTP (Stream Control Transmission Protocol) parsing and classification.
    - Replace parameter “ enCustUdp” with “ sctpPort” where the “ enCustUdp” is no longer required by the enhanced custom LUT2 operation.

  To maintain backward compatibility, set “ sctpPort” to 0. It is not required to update the tables with multiple paIpInfo_t entries unless “ enCustUdp” is used.

- Packet routing configuration data structure paRouteInfo_t is enhanced to support enhanced custom lookup operation, SRIO routing and optional post-classification command processing by adding following parameters:
    - Custom Type and Custom Index
    - SRIO packet type
    - Optional command pointer

  To maintain backward compatibility, all the new parameters should be set to 0. If there are some tables with one or more paRouteInfo_t entries in the application, it is desired to add four extra 0s into each entry to avoid unexpected behavior.

- Clarify the packet destination definitions for “ continue parse” by replacing

  “ pa_DEST_CONTINUE_PARSE” and

  “ pa_DEST_CONTINUE_PARSE_C1” with the following two definitions:
    - pa_DEST_CONTINUE_PARSE_LUT1: Packet remains in PA sub-system for more parsing and LUT1 classification
    - pa_DEST_CONTINUE_PARSE_LUT2: Packet remains in PA sub-system for more parsing and LUT2 classification

  To maintain backward compatibility, replace
  “ pa_DEST_CONTINUE_PARSE_C1” with
  “ pa_DEST_CONTINUE_PARSE_LUT1” and replace
  “ pa_DEST_CONTINUE_PARSE” with
  “ pa_DEST_CONTINUE_PARSE_LUT2” if the next classification is UDP/TCP/GTP-U (LUT2).

- The command buffer size requirement definition group cmdMaxBufSize has been renamed to cmdMinBufSize which defines the minimum buffer size required to contain the configuration command packets as defined below:
    - pa_ADD_MAC_MIN_CMD_BUF_SIZE_BYTES
    - pa_DEL_HANDLE_MIN_CMD_BUF_SIZE_BYTES
    - pa_DEL_L4_HANDLE_MIN_CMD_BUF_SIZE_BYTES
    - pa_ADD_IP_MIN_CMD_BUF_SIZE_BYTES
    - pa_ADD_PORT_MIN_CMD_BUF_SIZE_BYTES
    - pa_CONFIG_EXCEPTION_ROUTE_MIN_CMD_BUF_SIZE_BYTES
    - pa_REQUEST_STATS_MIN_CMD_BUF_SIZE_BYTES

Release 1.1.0.5:

- PA examples have compile option: SIMULATOR_SUPPORT. In case if example needs to be executed in simulator please compile with define SIMULATOR_SUPPORT. Alternatively set following variables at run time:
    - cpswSimTest = 0
    - cpswLpbkMode = CPSW_LOOPBACK_PA
- PA examples have compile option: SIMULATOR_SUPPORT. In case if example needs to be executed in simulator please compile with define SIMULATOR_SUPPORT.
- Added support for additional PDK packages
- Enhanced all examples and unit tests to run on both the simulator and the real silicon.

Release 1.1.0.4:

- Support C66 ELF
- Added PA LLD version related APIs
- Update PA examples and Unit Tests per Keystone C6616 PDK 1.0.0.9

Release 1.1.0.3:

- Support both COFF and ELF
- Update PA examples and Unit Tests per Keystone C6616 PDK 1.0.0.8

Release 1.1.0.2:

- Support ELF

Release 1.1.0.1:

- Added EMAC test example
- Added build option to support both Keystone C6616 and C6608 release package
- Resolved IRs

IR Parent/ Child Number	Severity Level	IR Description
00068852	Major	PA Firmware Error on QT (Burst packets handling in PA)

Release 1.1.0.0:

- Modified types from XDC to C99
- Changed all source, header, and example code to reflect CSL include path change in Keystone C6616 CSL version 1.0.0.14
- Changed XDC tool version to 3.16.02.32 in example and test projects
- Modified the PA LLD APIs to remove the TI XDIAS dependency
    - Pa_numAlloc() is removed. The constant pa_N_BUFS should be used to define the number of memory buffers required by PA LLD instead.
    - Pa_alloc (const IALG_Params*, struct IALG_Fxns **, IALG_MemRec *) is replaced with the new API Pa_getBufferReq(paSizeInfo_t *, int size[], int align[]).
        - The ILAG_Params type of configuration information is replaced by paSizeInfo_t.
        - The optional struct IALG_Fxns pointer is removed
        - The IALG_MemRec is replaced with the memory size and alignment requirement arrays.
    - Pa_init (IALG_Handle , const IALG_MemRec *, IALG_Handle , const IALG_Params *) is replaced with the new API Pa_create (paConfig_t *cfg, void* bases[], Pa_Handle *pHandle).
        - The ILAG_Params type of configuration information is replaced by paConfig_t.
        - This API should be called with the allocated memory buffer base addresses in stead of the IALG_MemRec.
        - This API will return the PA LLD handle which identifies the PA LLD instance and should be used for all other PA LLD API calls.
    - Pa_free (IALG_Handle , IALG_MemRec *) is replaced with the new API Pa_close (Pa_Handle handle, void* bases[])
        - This function returns the allocated memory buffer base addresses in array bases[] so that the application can free the buffers.
- Modified the PA LLD to remove unnecessary source level dependencies
    - Replace the local CSL file src/cslr_pass.h with the cslr_pa_ss.h in the CSL package.
    - Move the PA system statistics related definitions from src/pafrm.h to pa.h and rename sysStats_t to paSysStats_t.
        - The application code no longer needs to include “src/pafrm.h”
        - Need to search and replace sysStats_t with paSysStats_t at the application code.
    - Rename pasahost_temp.h to pasahost.h
        - Need to search and replace “pasahost_temp.h” with “pasahost.h”
    - The application no longer needs to declare the global variable passRegs.

- **Resolved IRs**

IR Parent/ Child Number	Severity Level	IR Description
00068460	Critical	PA firmware not accepting the ARP ethertype
00068119	Minor	PA LLD to use PA CSL from the CSL package
00067721	Major	Remove dependency of external variables
00061217	Minor	Big endian library mis-named

Release 1.0.0.7:

- Modifications to the examples and unit tests to be compatible with the latest CPPI and QMSS LLD (version 1.0.0.5).

Release 1.0.0.6:

- Modifications to the PDSP firmware to support the latest Keystone C6616/C6608 simulator (0.8.0)
- Added Custom Lookup support

Release 1.0.0.5:

- Modifications to the examples and unit tests to support the new CPPI specification (4.2.9)

Release 1.0.0.4:

- Internal Release only

Release 1.0.0.3:

- Modifications to the examples and unit tests to support the new CPPI specification (4.2.7)

Release 1.0.0.2

Release 1.0.0.1:

- Internal Release only

Release 1.0.0.0:

- Initial Release

Licensing

Please refer to the software Manifest document for the details.

Delivery Package

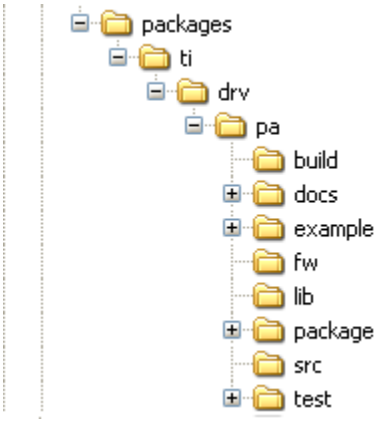
There is no separate delivery package. The PA LLD is being delivered as part of PDK within the MCSDK.

Installation Instructions

The LLD is currently bundled as part of Platform Development Kit (PDK) within the MCSDK. Refer installation instruction to the release notes provided for PDK.

Directory structure

After installation the PA LLD has the following directory structure:



The following table explains each individual directory:

Directory Name	Description
`ti/drv/pa`	The top level directory contains the following:- 1. *Environment configuration batch file* The file "`setupenv.bat`" is used to configure the build environment for the PA low level driver. 2. *XDC Build and Package files* These files (`config.bld, package.xdc` etc) are the XDC build files which are used to create the PA package. 3. *Exported Driver header file* Header files which are provided by the PA low level driver and should be used by the application developers for driver customization and usage.
`ti/drv/pa/build`	The directory contains internal XDC build related files which are used to create the PA low level driver package.
`ti/drv/pa/docs`	The directory contains the PA low level driver documentation.
`ti/drv/pa/example`	The "`example`" directory in the PA low level driver contains a simple example and an EMAC example.
`ti/drv/pa/test`	The "test" directory in the PA low level driver contains various unit tests
`ti/drv/pa/fw`	C data files required to configure the PA hardware sub-system.
`ti/drv/pa/lib`	The "`lib`" folder has pre-built Big and Little Endian libraries for the PA low level driver along with their *code/data size information*.
`ti/drv/pa/package`	Internal PA low level driver package files.

`ti/drv/pa/src`	Source code for the PA low level driver.

Customer Documentation List

Table 4 lists the documents that are accessible through the **/docs** folder on the product installation CD or in the delivery package.

Table 4 Product Documentation included with this Release

Document #	Document Title	File Name
1	API documentation (generated by Doxygen)	docs/paDocs.chm
2	Release Notes (this document)	docs/ReleaseNotes_PA_LLD.pdf
3	Software Manifest document	docs/PA_LLD_SoftwareManifest.pdf