Release Notes
For
Switch Software Development Kit

SDK 6.5.12

Table of Contents

Section 1: About This Document

This document provides a general description of the release and its new features. It also describes the chips supported by the release, BCM API additions or changes, resolved issues, and any relevant open issues. The reader should refer to prior release notes for 6.5.x, as only new features or issues are described in this version of the release notes.

Section 2: Product Documentation

The following documents are available through Broadcom's Customer Support Portal at https://csp.broadcom.com/group/customers/csp. They are the primary source of information and should be referenced when using this release:

Document	***Description***
56XX-PG6512-R	*BCM API Reference Guide* This manual describes the theory of operations of the API and all existing BCM APIs for this release. Available on docSAFE per request.
SDK-PG822-R	*Network Switching Software Platform Guide* This guide describes the SDK source and Makefile structure, abstraction and porting layers, device specific interactions, and the platform/operating system specific features of the SDK. If this is your first time working with the SDK, start with this document. Available through SDS Software Request Portal - must be requested.
56XX-PG-1001-R	*Network Switching SDK CINT Interface for Diagnostic Shell* This guide describes how to use the C interpreter (CINT) that runs under the diagnostic shell (Broadcom Shell utility). Available on docSAFE per request.
StrataXGS-AN300-R	*BCM Diagnostic Shell* This guide describes how to use the diagnostic shell, the primary CLI to the SDK. Available on docSAFE per request.

Additionally, starting with this release, please review the RN-SDK65xDNX-R document for DNX Release Notes for SDK 6.5.x. This is a companion guide describing only specific DNX family device changes in this SDK release. Common changes and resolved issues are described within this document which is packaged in the release deliverable itself.

Broadcom engineering values our customers' feedback on our releases and release notes documentation. In order to improve the material and provide customers with the most pertinent information, please follow this survey link to provide your feedback on the current release. Alternatively, you may email ron.meadows@broadcom.com with any free form feedback as well. We will consider implementing your suggestions for improvement in future versions of our releases and documentation.

Section 3: New Devices added to this release

For any given SDK release, support for certain devices may be provided in Preview or Supported status. Devices in "Supported Switch Devices" have completed the full QA process and are intended for use in production systems. It is expected that customers would integrate the version of the SDK which provides "Supported" status for their use on actual development or production systems. For the full list of Broadcom switch and PHY devices supported in the SDK, please reference the file `SDK-6.5.12-Device-Matrix.xlsx` in the RELDOCS directory in the release package.

Devices in "Preview Switch Devices" are provided to allow early integration of the customer's application with the SDK APIs that support that device. This software has not been fully tested on the physical target device and should not be expected to fully function.

Section 3.1: Newly Supported XGS Switch Devices in this release

Family	*Devices*	*Description*
BCM56870	BCM56870 A0	3200 Gb/s Switch Fabric 128x 25GbE/64x 50GbE/32x 100GbE Multilayer Switch
	BCM56873 A0	48x10/25GbE + 8x100GbE RoHS 6/6-compliant
BCM56970	BCM56972 B0	6.4 Tbps Multilayer Switch
	BCM56975 B0	6.4 Tbps Multilayer Switch
BCM53570	BCM53570 B0	High Port Count Integrated Switch With 100-FX/1G/2.5G/5G/10G/25G-Capable SerDes Lanes
	BCM53575 B0	8x1G + 4x10G
BCM56170	BCM56170 B0	High Port Count Integrated Switch With 100-FX/1G/2.5G/5G/10G/25G-Capable SerDes Lanes
	BCM56172 B0	24xMG-lite + 4x25G + 2x40G/8x10G
	BCM56174 B0	24X1G + 4x10G + 2x40G/8x10G
BCM53547	BCM53547 A0	24-Port plus 4 uplink GbE Switch w/ Max 8x2.5G Ports, and 24 Integrated PHYS, and CPU

	BCM53548 A0	16-Port plus 4 uplink GbE Switch w/ Max 8x2.5G Ports, and 16 Integrated PHYS, and CPU
	BCM53549 A0	8-Port plus 4 uplink GbE Switch w/ Max 8x2.5G Ports, and 8 Integrated PHYS, and CPU

Section 3.2: Preview XGS Switch Devices

Family	*Devices*	*Description*
No new preview switch devices are supported in this release		

Section 4: New Features per Device

Section 4.1: BCM56870 (TRIDENT3 X7) FAMILY GA (GENERAL AVAILABILITY) SUPPORT

The Broadcom® BCM56870 family is a class of high-performance, non-blocking network switching devices supporting up to a maximum of 128x 25GbE, 64x 50GbE, or 32x 100GbE, as well as various combinations of these port configurations. The BCM56870 delivers high-bandwidth, glueless network connectivity up to 3.2 Tbps on a single chip. In this release, support is provided for the BCM56870 A0 device specifically for the following configurations:

- 32 x 100GbE
- 64 x 50GbE
- 64 x 40GbE
- 128 x 25GbE
- 128 x 10GbE

Section 4.1.1: Status of SDK advanced feature support

In this release all features are of GA quality except for these advanced features listed in the below table:

Table 4. Advanced Feature Status

Description	*BCM56870 A0 Status*
ISSU	Not supported
In-Band Flow Analyzer	Beta
Flow Tracker	Beta

Section 4.1.2: BCM56870 new feature status

The SDK now provides generic APIs to enable the following newly supported flows. The user can create virtual ports to initiate and terminate tunnels for these flows. The virtual ports created for these flows can be members of the same forwarding domain (L2 VPN).

To enable the new flow support, set `flow_init_mode=1` in the config.td3.bcm. When this configuration is set, classic VxLAN and L2GRE BCM APIs will return not available. Use the new flow APIs to program VxLAN and L2GRE flows. Refer to the API guide for details of the new flow APIs and other APIs supported in this release. Please also refer to KB0026608 for an example of how to use the new flow APIs programming VxLAN flows.

The table below shows the new features supported through the new flow APIs:

Table 5. Status of advanced SDK features

Description	***BCM56870 A0 Status***
INT	GA
Classic VxLAN	GA
IPv6 VxLAN	GA
Classic L2GRE	GA
NSH	GA
GENEVE	GA
GPE	GA
MPLS over GRE	GA

Section 4.1.3: InBand Telemetry

In-band Telemetry refers to the framework where telemetry data is embedded inside the dataplane packet itself thereby enabling higher granularity and scalable network monitoring. There are number of implementation mechanisms for in-band telemetry and Trident 3 supports 3 methods: Inband-OAM, Lapukhov draft, and Tail stamping.

The egress port arrival rate and egress port drop byte counter will be populated in a future release with enhanced SDK InBand Telemetry support.

Section 4.1.4: M0 Firmware

In this release, firmware based LED processing and firmware based linkscan are supported through a new M0 firmware load for the BCM56870 device.

Two binary files `rc/cmicfw/linkscan_led_fw.bin` and `rc/cmicfw/custom_led.bin` support this feature. Support for loading these files into M0 memory is available via new diag shell command “M0 LOAD:”

```
M0 load 0 0x0 linkscan_led_fw.bin

M0 load 0 0x3800 custom_led.bin
```

Both of these files must be loaded for proper functioning of firmware based linkscan and LED processing. LED and linkscan specific diag commands are compatible with this new scheme except `led load` and `led prog` commands are replaced with “`M0 load`” command in BCM56870.

Section 4.1.5: PCIe Firmware

PCIe Gen3 FW v2.1 is being released along with this SDK release. PCIe Gen3 FW v2.1 fixes several issues related to PCIe link stability and it is recommended to upgrade BCM56870 device based systems with PCIe Gen3 FW v2.1 to establish stable PCIe link between BCM56870 and control CPU. Further details and upgrade instructions can be found in PCIe Gen3 Fw v2.1 release package.

Section 4.2: BCM56170 / BCM53570 FAMILY (HURRICANE3-MG/GREYHOUND2/QUARTZ) B0 GA (GENERAL AVAILABILITY) SUPPORT

The Broadcom® BCM56170 and BCM53570 System-on-a-Chip (SoC) series devices feature an integrated high-speed ARM Cortex-A9 processor, embedded Cortex-R5 processor and enterprise-level buffer and table sizes. This release provides GA support for BCM56170 B0, BCM56172 B0, BCM56174 B0, BCM53570 B0, and BCM53575 B0 devices.

The following configurations are supported in this release:

- Option 1: 24x10G OR 6x40G + 4x10G OR 1x40G + 4x25G OR 2x50G
- Option 2: 24x2.5G + 32x10G OR 8x40G
- Option 2a: 32x10G OR 8x40G + 30G RioT
- Option 3a: 12x10G + 36x1G + 20G RioT + 4x25G OR 2x50G + 8x10G OR 2x40G
- Option 4a: 48x2.5G + 20G RioT + 4x25G OR 2x50G + 8x10G OR 2x40G
- Option 5: 52x2.5G + 4x25G OR 2x50G + 8x10G OR 2x40G
- Option 6: 48x1G + Flex XAUI + 4x25G OR 2x50G + 8x10G OR 2x40G
- Option 7: 32x10G OR 8x40G + 4x1G
- Option 7b: 24x10G OR 6x40G + 40G RioT
- Option 8b: 24x2.5G + 40G RioT + 4x25G OR 2x50G + 8x10G OR 2x40G
- Option 9b: 48x1G + 40G RioT + 4x25G OR 2x50G + 8x10G OR 2x40G
- Option 10b: 32x1G (SGMII) + 40G RioT + 4x25G OR 2x50G + 8x10G OR 2x40G
- Option 11b: 32x1G (QSGMII) + 40G RioT + 4x25G OR 2x50G + 8 10G OR 2x40G
- Option 12: 24x10G OR 6x40G + 4x1G
- Option 13: 24x1G (SGMII) + 4x10G + XAUI

Section 4.2.1: Features supported

In this release all features are of GA quality.

Table 7. Supported Feature Status

Description	*BCM5617x B0/ BCM5357x B0*	*Notes*
Legacy Features		
COSQ	GA	
E2ECC	GA	
E2EFC	GA	
Field	GA	
IPMC	GA	BCM5617x only
KNET	GA	
L2	GA	
L3	GA	BCM5617x only
Link	GA	
Mcast	GA	
MiM Lite	GA	
Mirror	GA	
Multicast	GA	
NIV	GA	
OAM	GA	
Packet Tx/Rx	GA	
Policer	GA	
Port	GA	
Port Extender	GA	
QoS	GA	
Rate	GA	
SER	GA	
Stack	GA	
Stat	GA	
STG	GA	
Switch Controls	GA	
Time/Timesync	GA	
Trunk	GA	
uRPF	GA	BCM5617x only
VLAN	GA	
Warmboot	GA	
WRED	GA	

Section 4.2.2: New and enhanced features supported on BCM56170 and BCM53570 B0

The following table shows the maturity level of new and enhanced features in this SDK release.

Table 4. New Feature Status

Description	***BCM5617x B0/ BCM5357x B0***	***Notes***
2-Stage Schedulers	GA	
L2 My Station TCAM	GA	
L3 VRF	GA	BCM5617x only
RRoCE	GA	BCM5617x only
RTAG7	GA	
TSN-Preemption	GA	
TSN-Seamless Redundancy	GA	
TSN-Service Metering	GA	
TSN-TAS	GA	
TSN-TAF 802.1Qci	GA	
VXLAN Lite	GA	BCM5617x only
RioT	GA	BCM5617x only

Section 4.2.3: Notes and limitations

- The IPMC behavior after VxLAN packet decapsulation is different from the normal IPMC behavior. With normal IPMC behavior, the packet's VLAN and MAC_SA will be modified by EGR_L3_INTF added in the multicast group, but when VxLAN packet is being de-capsulated, it will use its payload VLAN and MAC_SA to overwrite the previous logic which makes it different from the normal IPMC behavior. The EGR_L3_INTF related parameters for creating a IPMC replication become dummy parameters, only to supply the needed API inputs to create destination port for the multicast group.

Section 4.3: BCM53547 A0 (WOLFHOUND2) FAMILY GA (GENERAL AVAILABILITY) SUPPORT

The Broadcom® BCM53547 switch family offers industry-leading integration and low power in a small footprint. The device offers up to 24 GbE ports + four 1G/2.5G uplink ports in a 25mm x 25mm package. Offering the industry's highest level of integration, the BCM53547 has 24 embedded copper GPHYs, as well as a powerful ARM A9 processor. The BCM53547 is ideal for cost sensitive edge applications, such as light L2-managed wiring closet switches for enterprise.

In this release, the BCM53547 A0 device is supported with the following configurations:

- 20 x 1G_GPHY + 8 x 2.5G_SGMII (option 1)
- 24 x 1G_GPHY + 4 x 2.5G_SGMII (option 2)
- 22 x 1G_GPHY + 2 x 1G_SGMII + 4 x 2.5G_SGMII (option 3)
- 22 x 1G_GPHY + 6 x 1G_SGMII (option 4)
- 20 x 1G_GPHY + 8 x 1G_SGMII (option 5)
- 24 x 1G_GPHY + 4 x 1G_SGMII (option 6)

The BCM53548 A0 device is supported with the following configurations:

- 12 x 1G_GPHY + 8 x 2.5G_SGMII (option 7)
- 16 x 1G_GPHY + 4 x 2.5G_SGMII (option 8)
- 14 x 1G_GPHY + 2 x 1G_SGMII + 4 x 2.5G_SGMII (option 9)
- 14 x 1G_GPHY + 6 x 1G_SGMII (option 10)
- 12 x 1G_GPHY + 8 x 1G_SGMII (option 11)
- 16 x 1G_GPHY + 4 x 1G_SGMII (option 12)

The BCM53549 A0 device is supported with the following configurations:

- 4 x 1G_GPHY + 8 x 2.5G_SGMII (option 13)
- 8 x 1G_GPHY + 4 x 2.5G_SGMII (option 14)
- 6 x 1G_GPHY + 2 x 1G_SGMII + 4 x 2.5G_SGMII (option 15)
- 6 x 1G_GPHY + 6 x 1G_SGMII (option 16)
- 4 x 1G_GPHY + 8 x 1G_SGMII (option 17)
- 8 x 1G_GPHY + 4 x 1G_SGMII (option 18)

Section 4.3.1: Features supported

The following table shows the maturity level of legacy features in this SDK release.

Table 7. Supported Feature Status

Description	***BCM53547 A0 Status***
Legacy Features	
COSQ	GA
Field	GA
IPMC	GA
KNET	GA
L2	GA
L3	GA
Link	GA
Mcast	GA
Mirror	GA
Multicast	GA
NIV	GA
OAM	GA
Packet Tx/Rx	GA

Policer	GA
Port	GA
Port Extender	GA
QoS	GA
Rate	GA
SER	GA
Stack	GA
Stat	GA
STG	GA
Switch Controls	GA
Time/Timesync	GA
Trunk	GA
uRPF	GA
VLAN	GA
Warmboot	GA

Section 4.4: Updated BCM56970 (Tomahawk2) Family Support

Section 4.4.1: New SKU support

In this release, GA level support is available for BCM56972 B0 and BCM56975 B0 devices.

Section 4.5: Embedded Applications Updates

Section 4.5.1: Flow Tracker

Flow Tracker is an embedded application that runs on ARM core 0 of Tomahawk (BCM56960) at Beta quality in this release. The application is used to learn incoming flows mapped to a user defined Flow Group and periodically export user configurable information about the flows to an IPFIX collector. The application can also identify elephant flows and change their QoS parameters dynamically.

Currently the Flow Tracker application supports only IPv4 5-tuple (IPv4 source IP address, IPv4 destination IP address, L4 source port, L4 destination port, IP protocol) based learning. IPv6 flow tracking is not supported. Enabling Flow Tracker will prevent the user from using the exact match feature, and will also reserve some flex counter pools for the app.

Refer to Section 12 for details on supported firmware release and the API guide for more information on the application and its features.

Section 5: Things to note

This section lists items that require special attention that are new to this release. Please see prior 6.5.x release notes for previously reported items that should also be noted.

Section 5.1: SDK releases out of active engineering support

The following releases are out of active engineering support :

- SDK 6.5.x releases: 6.5.4, 6.5.3, 6.5.2, 6.5.1, 6.5.0
- All SDK 6.4.x, 6.3.x, and older releases

Customers are recommended to use this release for new product development or sustaining releases. The following table shows the number of issues and improvements that have been added to our supported SDK releases by device over the past 12 months. While the table shows individual devices, many issues and improvements will apply to multiple products, e.g. BCM56850 and BCM56860, or all XGS products.

Per Broadcom policy, as older devices are discontinued due to end of life (EOL), their SW support is also deprecated in SDK releases beyond the device EOL date. All releases earlier than SDK 6.2.1 and SDK 5.x.x are EOL.

Section 5.2: External PHY updates

In this release, there have been no significant feature enhancements made for external PHY driver support. Please see the Resolved Issues section for any new improvements done to external PHY driver code.

Section 5.3: Warmboot Notes and Considerations

This section is to give information about warmboot specific activity in this release. Please note that the warmboot scache size requirements for a device for a particular release can be found by running the `warmboot storage` command at the BCM prompt.

Section 5.3.1: Validated Warmboot upgrades

Warmboot testing and issue resolution is focused on a majority of recently supported devices including BCM56870 and is performed with a limited set of test cases. Warmboot testing is not performed with PHY devices attached. It is recommended that any customer perform their own warmboot testing for their specific environment and use these results as guidance only.

SDK warmboot upgrades from 6.5.11 to 6.5.12 have been validated on specific silicon validation kits (SVKs) in this release.

Section 5.3.2: Validated Warmboot downgrades

Broadcom has performed limited release downgrade testing from 6.5.12 to 6.5.11 using BCM56960 and BCM56850 family devices. Broadcom does not perform downgrade testing from release 6.5.x to release 6.4.x.

Warmboot downgrade is not supported on BCM88670, BCM56860, BCM56565, BCM56760, BCM56970, and BCM56870 device families due to portmod limitations.

Section 5.3.3: BCM56870 special considerations

BCM56870 is a programmable device and is released with CANCUN firmware. When rebooting with the same version of the CANCUN, warmboot is supported. However, in the future when CANCUN upgrade is performed, a cold boot is required to reprogram the device.

Section 5.3.4: BFD warmboot upgrade considerations

Customers performing warmboot upgrade from SDK 6.5.11 to SDK 6.5.12 may see an error invoking `bcm_bfd_status_multi_get()`. Please contact support for a patch referencing SDK-136418.

Section 5.4: Field data qualifiers update

The IFP data qualifier APIs `bcm_field_data_nn` are used to allocate and attach the UDF resources to Field module. UDF module APIs (`bcm_udf_nn`) have been introduced to make it independent of FP, since the UDF resources are also being used by other modules (Flex Hash, RTAG7).

Going forward, users must use only UDF module APIs to allocate UDF resources. Support has been removed for all the field data qualifier APIs for BCM56870 and all future new devices. These APIs are still supported for legacy devices (prior to BCM56870).

Section 5.5: BCM56870 multiple unit control

Customers who deploy a single CPU controlling multiple BCM56870 units, please contact FAE/AE support to understand the configuration details further and provide any necessary patches.

Section 6: Summary of BCM API changes and enhancements

This section summarizes BCM API changes in this release. Complete documentation will be available in the Network Switching Software Programmer's Guide number 56XX-PG6512-R.

For the full list of API support by Broadcom device, please reference the file SDK-6.5.x-Support-Matrix.xls in the sdk/RELDOCS directory in the release package. The API support matrix is not maintained for DNX devices, thus DNX devices are excluded from SDK-6.5.x-Support-Matrix.xls.

Broadcom does not guarantee API default values set within the SDK and changes to default values may be made between releases. If an API default value is required for application software to work properly, it must be explicitly set.

Section 6.1: Class of Service Queue Configuration API

Table: Features that can be controlled on a gport/cosq basis.

CosqStatOBMLossless1EnqueuedBytes	OBM Lossless Class 1 enqueued bytes count.
CosqStatOBMLossyHighPriEnqueuedPackets	OBM high priority enqueued packets count.
CosqStatOBMLossyHighPriEnqueuedBytes	OBM high priority enqueued bytes count.
CosqStatOBMLossyLowPriEnqueuedPackets	OBM low priority enqueued packets count.
CosqStatOBMLossyLowPriEnqueudBytes	OBM low priority enqueued bytes count.
CosqStatOBMExpressEnqueuedPackets	OBM express enqueued packets count.
CosqStatOBMExpressEnqueuedBytes	OBM express enqueued bytes count.
CosqStatOBMExpressDroppedPackets	OBM express dropped packets count.

CosqStatOBMExpressDroppedBytes	OBM express dropped bytes count.
CosqStatMaxCount	this should be the last one.

Table: CoSQ Control Type Values

Value	*Description*	*Arg value*
162	bcmCosqControlIngressPortPGResetOffsetBytes	Ingress port PG reset Offset setting
163	bcmCosqControlIngressPublicPoolLimitBytes	Public Service Pool Limit Bytes

```
#define BCM_COSQ_BW_BURST_CALCULATE        0x00000800 /* Re-calculate burst
                                                         based on configured
                                                         rate. */
```

Section 6.2: Fabric

Table: Fabric Type Values

Value	*Description*	*Arg Value*
bcmFabricDelaySingleCellInFabricRx	Enable/Disable delay of single cell in fabric rx, to improve bus utilization	1/0

```
typedef enum fabric_cgm_control_type_e {
    FabricCgmRciLinkPipeLevelTh = 0,/* RCI controls */
    FabricCgmRciEgressPipeLevelTh = 1,
    FabricCgmRciTotalEgressPipeLevelTh = 2,
    FabricCgmRciLocalLevelTh = 3,
    FabricCgmRciTotalLocalLevelTh = 4,
    FabricCgmRciLinkLevelFactor = 5,
    FabricCgmRciEgressLevelFactor = 6,
    FabricCgmRciHighSeverityMinLinksParam = 7,
    FabricCgmRciHighSeverityFactor = 8,
    FabricCgmRciCoreLevelMappingTh = 9,
    FabricCgmRciDeviceLevelMappingTh = 10,
    FabricCgmNof = 11,
} fabric_cgm_control_type_t;
```

bcm_fabric_cgm_control_set bcm_fabric_cgm_control_get

Configure Fabric CGM.

SYNOPSIS

```
#include <bcm/fabric.h>
int
bcm_fabric_cgm_control_set(
    int unit,
    uint32 flags,
    bcm_fabric_cgm_control_type_t control_type,
    bcm_fabric_cgm_control_id_t control_id,
    int value);
```

```
int
bcm_fabric_cgm_control_get(
    int unit,
    uint32 flags,
    bcm_fabric_cgm_control_type_t control_type,
    bcm_fabric_cgm_control_id_t control_id,
    int * value);
```

PARAMETERS

unit	(IN) Unit number.
flags	(IN) Flags
control_type	(IN)
control_id	(IN)
value	(IN) / (OUT) Value

DESCRIPTION

Configure Fabric CGM.

RETURNS

BCM_E_xxx

Section 6.3: Kernel Network (KNET) Configuration

```
typedef struct knet_filter_s {
    int id; /* Filter ID. */
    int type; /* Filter type (BCM_KNET_FILTER_T_XXX). */
    uint32 flags; /* BCM_KNET_FILTER_F_XXX flags. */
    int priority; /* Filter priority (0 is highest). */
    int dest_type; /* Filter destination type. */
    int dest_id; /* Filter destination ID. */
    int dest_proto; /* If non-zero this value overrides the default protocol type when matching packet
is passed to network stack. */
    int mirror_type; /* Mirror destination type. */
```

```
    int mirror_id; /* Mirror destination ID. */
    int mirror_proto; /* If non-zero this value overrides the default protocol type when matching
packet is passed to network stack. */
    uint32 match_flags; /* BCM_KNET_FILTER_M_XXX flags. */
    char desc; /* Filter description (optional) */
    bcm_vlan_t m_vlan; /* VLAN ID to match. */
    bcm_port_t m_ingport; /* Local ingress port to match. */
    int m_src_modport; /* Source module port to match. */
    int m_src_modid; /* Source module ID to match. */
    bcm_rx_reasons_t m_reason; /* Copy-to-CPU reason to match. */
    int m_fp_rule; /* Filter processor rule to match. */
    int raw_size; /* Size of valid raw data and mask. */
    uint8 m_raw_data; /* Raw data to match. */
    uint8 m_raw_mask; /* Raw data mask for match. */
    uint32 cb_user_data; /* Filter user data for knet rx cb. */
} knet_filter_t;



typedef struct knet_netif_s {
    int id; /* Network interface ID. */
    int type; /* Network interface type (BCM_KNET_NETIF_T_XXX). */
    uint32 flags; /* BCM_KNET_NETIF_F_XXX flags. */
    bcm_mac_t mac_addr; /* MAC address associated with this network interface. */
    bcm_vlan_t vlan; /* VLAN ID associated with this interface. */
    bcm_port_t port; /* Local port associated with this interface. */
    bcm_cos_queue_t cosq; /* Cos offset from base queue of port associated with this interface. */
    char name; /* Network interface name (assigned by kernel) */
    uint32 cb_user_data; /* Netif user data for knet rx cb. */
} knet_netif_t;
```

Section 6.4: Layer 2 Address Management

```
/* Delete a MAC entry according to mac, vid (used as VSI) and modid. When modid != 0 it should
hold a VLAN ID for IVL entries. */
int l2_addr_by_struct_delete(
    int unit,
    bcm_l2_addr_t l2addr
)



/* Retrieve a MAC entry according to mac, vid (used as VSI) and modid. When modid != 0 it should
hold a VLAN ID for IVL entries. */
int l2_addr_by_struct_get(
    int unit,
    bcm_l2_addr_t l2addr
```

```
)


typedef struct l2_addr_s {
    uint32 flags;         /* BCM_L2_xxx flags. */
    uint32 flags2;        /* BCM_L2_FLAGS2_xxx flags. */
    uint32 station_flags; /* BCM_L2_STATION_xxx flags. */
    bcm_mac_t mac;        /* 802.3 MAC address. */
    bcm_vlan_t vid;       /* VLAN identifier. */
    int port;             /* Zero-based port number. */
    int modid;            /* XGS: modid. */
    bcm_trunk_t tgid;     /* Trunk group ID. */
    bcm_cos_t cos_dst;    /* COS based on dst addr. */
    bcm_cos_t cos_src;    /* COS based on src addr. */
    bcm_multicast_t l2mc_group;
                          /* XGS: index in L2MC table.
                             For SBX chips it is the Multicast
                             Group index */
    bcm_if_t egress_if;   /* XGS: index in Next Hop Tables.
                             Used it with BCM_L2_FLAGS2_ROE_NHI
                            flag */
    bcm_pbmp_t block_bitmap; /* XGS: blocked egress bitmap. */
    int auth;             /* Used if auth enabled on port. */
    int group;            /* Group number for FP. */
    bcm_fabric_distribution_t distribution_class;
                          /* Fabric Distribution Class. */
    int encap_id;         /* out logical interface */
    int age_state;        /* Age state of the entry */
    uint32 flow_handle;   /* FLOW handle for flex entries. */
    uint32 flow_option_handle;
                          /* FLOW option handle for flex entries. */
    bcm_flow_logical_field_t logical_fields;
                          /* logical fields array for flex entries. */
    uint32 num_of_fields; /* number of logical fields. */
    bcm_pbmp_t sa_source_filter_pbmp;
                          /* Source port filter bitmap for this SA */
    bcm_tsn_flowset_t tsn_flowset;
                          /* Time-Sensitive Networking: associated
                             flow set */
    bcm_tsn_sr_flowset_t sr_flowset;
                          /* Seamless Redundancy: associated flow set */
    bcm_policer_t policer_id; /* Base policer to be used */
    bcm_tsn_pri_map_t taf_gate_primap;
                          /* TAF (Time Aware Filtering) gate priority mapping */
} l2_addr_t;


typedef struct l2_station_s {
    uint32 flags;             /* BCM_L2_STATION_xxx flags. */
    int priority;             /* Entry priority. */
    bcm_mac_t dst_mac;        /* Destination MAC address to match. */
    bcm_mac_t dst_mac_mask;   /* Destination MAC address mask value. */
```

```
    bcm_vlan_t vlan;          /* VLAN to match. */
    bcm_vlan_t vlan_mask;     /* VLAN mask value. */
    bcm_port_t src_port;      /* Ingress port to match. */
    bcm_port_t src_port_mask; /* Ingress port mask value. */
    bcm_tsn_pri_map_t taf_gate_primap;
                          /* TAF (Time Aware Filtering) gate
                             priority mapping */
} l2_station_t;
```

Section 6.5: L3 interface APIs

bcm_l3_intf_qos_t_init

Initialize a bcm_l3_intf_qos_t structure.

SYNOPSIS

```
#include <bcm/l3.h>
void bcm_l3_intf_qos_t_init(bcm_l3_intf_qos_t *intf_qos);
```

PARAMETERS

intf_qos	(OUT) L3 interface QOS setting structure.

DESCRIPTION

Initialize a L3 interface QOS structure to default values. The function is for initializing a L3 interface QOS structre prior to filling it out and passing it to APIs.

RETURNS

Nothing

Section 6.6: Mirroring

Table: BCM Mirror Destination Flags

Name	*Description*
BCM_MIRROR_DEST_TUNNEL_WITH_SEQ	Include Sequence number in header (used with BCM_MIRROR_DEST_TUNNEL_IP_GRE)
BCM_MIRROR_DEST_UPDATE_EXT_COUNTERS	Set statistic interfaces for mirrored packets
BCM_MIRROR_DEST_FLAGS2_TUNNEL_VXLAN	Mirrored packet should be VXLAN Tunneled.

Table: BCM MIRROR EXT flags

Name	*Description*
BCM_MIRROR_EXT_STAT_ID_COUNT	Number of statistic interfaces for mirrored packets.

```
typedef struct mirror_destination_s {
    bcm_gport_t mirror_dest_id; /* Unique mirror destination and encapsulation identifier. */
    uint32 flags; /* See BCM_MIRROR_DEST_xxx flag definitions. */
    bcm_gport_t gport; /* Mirror destination. */
    uint8 version; /* IP header version. */
    uint8 tos; /* Traffic Class/Tos byte. */
    uint8 ttl; /* Hop limit. */
    bcm_ip_t src_addr; /* Tunnel source ip address (IPv4). */
    bcm_ip_t dst_addr; /* Tunnel destination ip address (IPv4). */
    bcm_ip6_t src6_addr; /* Tunnel source ip address (IPv6). */
    bcm_ip6_t dst6_addr; /* Tunnel destination ip address (IPv6). */
    uint32 flow_label; /* IPv6 header flow label field. */
    bcm_mac_t src_mac; /* L2 source mac address. */
    bcm_mac_t dst_mac; /* L2 destination mac address. */
    uint16 tpid; /* L2 header outer TPID. */
    bcm_vlan_t vlan_id; /* Vlan id. */
    bcm_trill_name_t trill_src_name; /* TRILL source bridge nickname */
    bcm_trill_name_t trill_dst_name; /* TRILL destination bridge nickname */
    int trill_hopcount; /* TRILL hop count */
    uint16 niv_src_vif; /* Source Virtual Interface of NIV tag */
    uint16 niv_dst_vif; /* Destination Virtual Interface of NIV tag */
    uint32 niv_flags; /* NIV flags BCM_MIRROR_NIV_XXX */
    uint16 gre_protocol; /* L3 GRE header protocol */
```

```
    bcm_policer_t policer_id; /* policer_id */
    int stat_id; /* stat_id */
    int stat_id2; /* stat_id2 for second counter pointer */
    bcm_if_t encap_id; /* Encapsulation index */
    bcm_if_t tunnel_id; /* IP tunnel for encapsulation. Valid only if
BCM_MIRROR_DEST_TUNNEL_IP_GRE is set */
    uint16 span_id; /* SPAN-ID. Valid only if BCM_MIRROR_DEST_TUNNEL_WITH_SPAN_ID is
set */
    uint8 pkt_prio; /* L2 header PCP */
    uint32 sample_rate_dividend; /* The probability of mirroring a packet is: sample_rate_dividend
>= sample_rate_divisor ? 1 : sample_rate_dividend / sample_rate_divisor */
    uint32 sample_rate_divisor;
    uint8 int_pri; /* Internal Priority */
    uint16 etag_src_vid; /* Extended (source) port vlan id */
    uint16 etag_dst_vid; /* Extended (destination) port vlan id  */
    uint16 udp_src_port; /* UDP source port */
    uint16 udp_dst_port; /* UDP destination port */
    uint32 egress_sample_rate_dividend; /* The probability of outbound mirroring a packet from the
destination is sample_rate_dividend >= sample_rate_divisor ? 1 : sample_rate_dividend /
sample_rate_divisor */
    uint32 egress_sample_rate_divisor;
    uint8 recycle_context; /* recycle context of egress originating packets */
    uint16 packet_copy_size; /*  If non zero, the copied packet will be truncated to the first
packet_copy_size . Current supported values for DNX are 0, 64, 128, 192 */
    uint16 egress_packet_copy_size; /* If non zero and the packet is copied from the egress, the
packet will be truncated to the first packet_copy_size . Current supported values for DNX are 0,
256. */
    bcm_mirror_pkt_header_updates_t packet_control_updates;
    bcm_mirror_psc_t rtag; /* specify RTAG HASH algorithm used for shared-id mirror destination */
    uint8 df; /* Set the do not fragment bit of IP header in mirror encapsulation */
    uint8 truncate; /* Setting truncate would result in mirroring a truncated frame  */
    uint16 template_id; /* Template ID for IPFIX HDR */
    uint32 observation_domain; /* Observation domain for IPFIX HDR */
    uint32 is_recycle_strict_priority; /* Recycle priority (1-lossless, 0-high) */
    int ext_stat_id; /* ext_stat_id to support statistic interface for mirror packets. */
    uint32 flags2; /* See BCM_MIRROR_DEST_FLAGS2_xxx flag definitions. */
    uint32 vni; /* Virtual Network Interface ID. */
    uint32 gre_seq_number; /* Sequence number value used in GRE header. If no value is
provided, gre_seq_number will start with 0. Valid only if BCM_MIRROR_DEST_TUNNEL_IP_GRE
is set. */
    bcm_mirror_pkt_erspan_encap_t erspan_header; /* ERSPAN encapsulation header fields. Valid
only if only BCM_MIRROR_DEST_TUNNEL_IP_GRE is set. */
} mirror_destination_t;



typedef struct mirror_pkt_erspan_encap_s {
    uint8 truncated_flag; /* If set to 0x1, the frame copy encapsulated in the ERSPAN packet will be
truncated. This should be set if ERSPAN encapsulated frame exceeds the configured MTU. */
    uint16 session_id; /* Identification associated with each ERSPAN session. */
    uint16 security_grp_tag; /* Security Group Tag of the monitored frame. */
    uint8 hw_id; /* Unique identifier of an ERSPAN engine within a system. */
    uint8 timestamp_granularity; /* If it is not set, NTP timestamp will be used. If it is set to 0x1, PTP
timestamp will used. */
```

```
    uint8 optional_hdr; /* Indicates if the optional platform-specific sub-header is present. */
    uint8 platform_id; /* Platform identifier to parse the optional platform-specific sub-header. Valid
only if optional_hdr is set. */
    uint16 virtual_supervisor_module_id; /* Identifier of Virtual Supervisor Module domain. Valid only
if platform_id set to 0x1. */
    uint16 switch_id; /* Identifies a source switch at the receiving end. Valid only if optional_hdr is
set and platform_id as 0x5 or 0x6. */
} mirror_pkt_erspan_encap_t;
```

Section 6.7: MPLS Management

Table: MPLS Port VCCV Type

Name	*Purpose*
bcmMplsPortControlChannelNone	No VCCV Channel
bcmMplsPortControlChannelAch	ACH Control Channel
bcmMplsPortControlChannelRouterAlert	MPLS Router Alert Channel
bcmMplsPortControlChannelTtl	MPLS PW Label with TTL = 1
bcmMplsPortControlChannelGalUnderPw	GAL label under MPLS PW Label

Table: MPLS Tunnel Switch Actions

Name	*Purpose*
BCM_MPLS_SWITCH_ACTION_SWAP	Incoming MPLS label is swapped. The egress_label parameter describes the outgoing MPLS label. The egress_intf points to an egress object (can be a normal port or MPLS tunnel).

Table: MPLS Egress Actions

Name	*Purpose*
BCM_MPLS_EGRESS_ACTION_SWAP	Incoming MPLS label is swapped.

```
/* Special Label Control Types. */
typedef enum bcm_mpls_special_label_push_type_e {
    bcmMplsSpecialLabelPushNone                     = 0, /* Nothing to push */
    bcmMplsSpecialLabelPushSpecial                  = 1, /* Push Special label */
    bcmMplsSpecialLabelPushEntropy                  = 2, /* Push Entropy label */
    bcmMplsSpecialLabelPushSpecialPlusEntropy       = 3  /* Push Entropy and Special labels */
} bcm_mpls_special_label_push_type_t;


/* Special Label Types. */
typedef enum bcm_mpls_special_label_type_e {
    bcmMplsSpecialLabelTypeNone                     = 0,
    bcmMplsSpecialLabelTypeGal                      = 1,
    bcmMplsSpecialLabelTypeRal                      = 2,
    bcmMplsSpecialLabelTypeSpecial                  = 3,
    bcmMplsSpecialLabelTypeEntropy                  = 4
} bcm_mpls_special_label_type_t;
```

Section 6.8: Operations, Administration, And Maintenance

```
typedef enum oam_upmep_pdu_type_e {
    OamUpmepPduTypeCcm = 0,/* Used to set the CPU queue for UP MEP CCM packets */
    OamUpmepPduTypeLmDm = 1,/* Used to set the CPU queue for UP MEP LM/DM packets */
    OamUpmepPduTypeSlowpath = 2,/* Used to set the CPU queue for UP MEP slowpath packets
*/
    OamUpmepPduTypeCount = 3,/* Must be last.  Not a usable value. */
} oam_upmep_pdu_type_t;
```

bcm_oam_upmep_cosq_get

Query device-wide UP MEP PDU type CPU queue

SYNOPSIS

```
#include <bcm/oam.h>
int bcm_oam_upmep_cosq_get(
    int unit,
    bcm_oam_upmep_pdu_type_t upmep_pdu_type,
    bcm_cos_queue_t *cosq);
```

PARAMETERS

unit	(IN) Unit number
upmep_pdu_type	(IN) UP MEP PDU type.
cosq	(OUT) pointer to CPU queue Value to be get for the UP MEP PDU type.

DESCRIPTION

Query device-wide UP MEP PDU type CPU queue

RETURNS

BCM_E_PARAM	Invalid parameter.
BCM_E_XXX	

bcm_oam_upmep_cosq_set

Configure device-wide UP MEP PDU type CPU queue

SYNOPSIS

```
#include <bcm/oam.h>
int bcm_oam_upmep_cosq_set(
    int unit,
    bcm_oam_upmep_pdu_type_t upmep_pdu_type,
    bcm_cos_queue_t cosq);
```

PARAMETERS

unit	(IN) Unit number
upmep_pdu_type	(IN) UP MEP PDU type.
cosq	(IN) CPU queue Value to be set for the UP MEP PDU type.

DESCRIPTION

Configure device-wide UP MEP PDU type CPU queue

RETURNS

BCM_E_PARAM	Invalid parameter.
BCM_E_XXX	

Section 6.9: Policer

```
typedef struct policer_group_info_s {
    bcm_policer_t policer_id; /* Policer Identifier. */
    bcm_policer_group_direction_t direction; /* Direction of allocation of entries */
    uint32 mode_id; /* Policer group mode identifier. */
    bcm_policer_group_mode_t mode; /* Policer group mode. */
    uint32 pool_id; /* Pool number of allocated entries. If direction is vertical, it represent pool in
which all entries are allocated. If direction is horizontal, it is always 0. */
    uint32 base_offset; /* In vertical allocation, it represents offset of Base policer Index from the
start of pool. In horizontal direction, it represents offset of each policers allocated from start of their
respective pool. */
    uint32 num_policers; /* Number of policers allocated in the group. */
    uint32 num_offset_reserved; /* Number of entries reserved in the hardware. Direction along with
this value provide information of how entries are allocated */
```

```
    bcm_policer_t macro_policer_id; /* Macro policer of the policer group. */
} policer_group_info_t;


typedef enum policer_group_direction_e {
    PolicerGroupDirectionVertical = 0,/* Entries allocation within pool */
    PolicerGroupDirectionHorizontal = 1,/* Entries allocation across pool */
} policer_group_direction_t;
```

bcm_policer_custom_group_id_copy

Copy a policer group from current hardware location to new location as provided.

SYNOPSIS

```
#include <bcm/policer.h>
int
bcm_policer_custom_group_id_copy(
    int unit,
    bcm_policer_group_info_t *old_policer_group_info,
    bcm_policer_group_info_t *new_policer_group_info);
```

PARAMETERS

unit	(IN) Unit number.
old_policer_group_info	(IN) Policer group info for old location.
new_policer_group_info	(IN/OUT) Policer group info for new location.

DESCRIPTION

When config property "global_meter_compaction_support" is set, this API is used to move policer group information from old locations to new locations.

RETURNS

BCM_E_XXX

bcm_policer_custom_group_id_create

Create a group of policers belonging to group mode Id. The output is policer group information required to build database of entries allocations within/across pools.

SYNOPSIS

```
#include <bcm/policer.h>
int
bcm_policer_custom_group_id_create(
    int unit,
    uint32 flags,
    uint32 mode_id,
    bcm_policer_t macro_flow_policer_id,
    bcm_policer_group_info_t *policer_group_info);
```

PARAMETERS

unit	(IN) Unit number.
flags	(IN) flags
mode_id	(IN) Created Mode Identifier for Policer Group Mode
macro_flow_policer_id	(IN) is an optional parameter and needs to be passed only while creating the micro flow policers of a hierarchical group (2 stage policers).
policer_group_info	(IN/OUT) policer group information.

DESCRIPTION

This API creates a group of policers belonging to customized group mode. bcm_policer_group_get must be called to get the list of policers created. bcm_policer_set must be called to setup the individual policers. The parameter macro_flow_policer_id needs to be set to "0" in case of 1 level policers. In case of 2 level policers, this parameter needs to be set with the value of second level policer while creating 1st level policers. The output policer group information contains information for building up database of entries allocation within or across pools in user application. This API is valid when config property "global_meter_compaction_support" is set.

RETURNS

BCM_E_XXX

bcm_policer_group_info_t_init

Initializes bcm_policer_group_info_t struct.

SYNOPSIS

```
#include <bcm/policer.h>
void bcm_policer_group_get(bcm_policer_group_info_t *policer_group_info);
```

PARAMETERS

policer_group_info	(IN/OUT) Pointer to struct bcm_policer_group_info_t to initialize.

DESCRIPTION

Initializes bcm_policer_group_info_t struct.

```
/* Direction of entries allocation in hardware. */
typedef enum bcm_policer_group_direction_e {
   bcmPolicerGroupDirectionVertical = 0, /* Entries allocation within pool */
   bcmPolicerGroupDirectionHorizontal = 1 /* Entries allocation across pool */
} bcm_policer_group_direction_t;

/* Policer group information */
typedef struct bcm_policer_group_info_s {
   bcm_policer_t policer_id;          /* Policer Identifier. */
   bcm_policer_group_direction_t direction; /* Direction of allocation of entries */
   uint32 mode_id;                  /* Policer group mode identifier. */
   bcm_policer_group_mode_t mode;      /* Policer group mode. */
   uint32 pool_id;                  /* Pool number of allocated entries. If
                                       direction is vertical, it represent
                                       pool in which all entries are
                                       allocated. If direction is
                                       horizontal, it is always 0. */
   uint32 base_offset;               /* In vertical allocation, it represents
                                       offset of Base policer Index from the
                                       start of pool. In horizontal
                                       direction, it represents offset of
                                       each policers allocated from start of
                                       their respective pool. */
   uint32 num_policers;              /* Number of policers allocated in the
                                       group. */
```

```
    uint32 num_offset_reserved;         /* Number of entries reserved in the
                                           hardware. Direction along with this
                                           value provide information of how
                                           entries are allocated */
    bcm_policer_t macro_policer_id;     /* Macro policer of the policer group. */
} bcm_policer_group_info_t;
```

RETURNS

None.

Section 6.10: Port Configuration

```
/* Get TX FIR parameters for a given port */
int port_phy_tx_get(
    int unit,
    bcm_port_t port,
    bcm_port_phy_tx_t tx
)


/* Modify TX FIR parameters for a given port */
int port_phy_tx_set(
    int unit,
    bcm_port_t port,
    bcm_port_phy_tx_t tx
)


/* Initialize the bcm_port_phy_tx_t structure. */
void port_phy_tx_t_init(
    bcm_port_phy_tx_t tx
)


typedef struct port_match_info_s {
    bcm_port_match_t match; /* Match criteria */
    bcm_gport_t port; /* Match port */
    bcm_vlan_t match_vlan; /* Outer VLAN ID to match */
    bcm_vlan_t match_vlan_max; /* Maximum VLAN ID in range to match */
    bcm_vlan_t match_inner_vlan; /* Inner VLAN ID to match */
    bcm_vlan_t match_inner_vlan_max; /* Maximum Inner VLAN ID in range to match */
    bcm_vlan_t match_tunnel_vlan; /* B-domain VID */
    bcm_mac_t match_tunnel_srcmac; /* B-domain source MAC address */
```

```
    bcm_mpls_label_t match_label; /* MPLS label */
    uint32 flags; /* BCM_PORT_MATCH_XXX flags */
    bcm_tunnel_id_t match_pon_tunnel; /* PON Tunnel value to match. */
    bcm_port_ethertype_t match_ethertype; /* Ethernet type value to match */
    int match_pcp; /* Outer PCP ID to match */
    bcm_vlan_action_set_t action; /* Match action */
    uint16 extended_port_vid; /* Extender port VID */
    bcm_vpn_t vpn; /* VPN ID */
    uint16 niv_port_vif; /* NIV port VIF */
    uint32 isid;
} port_match_info_t;


typedef struct port_phy_tx_s {
    int pre2;
    int pre;
    int main;
    int post;
    int post2;
    int post3;
    bcm_port_phy_tx_tap_mode_t tx_tap_mode;
    bcm_port_phy_signalling_mode_t signalling_mode;
} port_phy_tx_t;


/* Features that can be controlled on a per-port basis. */
typedef enum bcm_port_control_e {
  ...
  PortControlTXPISDMOverride  361  /* Override TXPI SDM value */
  PortControlTXPISDMStatus  362  /* The final result value of TXPI SDM */
  PortControlBicastPort  363  /* Sets the Bicast mapping for port */
  PortControlObmClassifierPriority  364  /* Indicates the default OBM Traffic class for port */
  PortControlObmClassifierEnableDscp  365  /* Enable OBM DSCP Classifier on port */
  PortControlObmClassifierEnableMpls  366  /* Enable OBM MPLS Classifier on port */
  PortControlObmClassifierEnableEtag  367  /* Enable OBM ETAG Classifier on port */
  PortControlObmClassifierEtagEthertype  368  /* Configure Ethertype for OBM ETAG Classifier */
  PortControlObmClassifierVntagEthertype  369  /* Configure Ethertype for OBM VNTAG Classifier
*/
} bcm_port_control_t;


typedef enum port_phy_signalling_mode_e {
    PortPhySignallingModeNRZ = 0,
    PortPhySignallingModePAM4 = 1,
} port_phy_signalling_mode_t;


typedef enum port_phy_tx_tap_mode_e {
    PortPhyTxTapMode3Tap = 0,
    PortPhyTxTapMode6Tap = 1,
```

```
} port_phy_tx_tap_mode_t;


/* New redirection packet color */
typedef enum bcm_port_redirect_color_e {
    bcmPortRedirectColorNone = 0,
    bcmPortRedirectColorGreen = 1,
    bcmPortRedirectColorYellow = 2,
    bcmPortRedirectColorRed = 3
} bcm_port_redirect_color_t;
```

API Port Configuration

bcm_port_resource_speed_multi_set

Modify the Port Speeds for a set of ports in a single Port Macro.

SYNOPSIS

```
#include <bcm/port.h>
int bcm_port_resource_speed_multi_set(
    int unit,
    int nport,
    bcm_port_resource_t *resource)
```

PARAMETERS

unit	BCM device number
nport	
resource	

DESCRIPTION

Modify the Port Speeds for a set of ports in a single Port Macro.

bcm_port_resource_speed_get
bcm_port_resource_speed_set

Get/Modify the speed for a given port.

SYNOPSIS

```
#include <bcm/port.h>
int bcm_port_resource_speed_get(
    int unit,
    bcm_gport_t port,
    bcm_port_resource_t *resource)

int bcm_port_resource_speed_set(
    int unit,
    bcm_gport_t port,
    bcm_port_resource_t *resource)
```

PARAMETERS

unit	BCM device number
port	
resource	

DESCRIPTION

Get/Modify the speed for a given port.

Section 6.11: SAT Overview

Section 6.11.1: SAT Gtf Bandwidth Info

SAT Gtf Bandwidth Info:

```
typedef struct bcm_sat_gtf_bandwidth_s {
    uint32 flags;
    uint32 rate;        /* traffic rate. Units: kbps */
    uint32 max_burst;   /* traffic burst. Units: kbit */
    uint32 granularity;  /* packet per second */
} bcm_sat_gtf_bandwidth_t;
```

Table: SAT GTF Rate Flag Definitions

BCM_SAT_GTF_RATE_IN_BYTES	If set, gtf rate is configured in bytes per second
BCM_SAT_GTF_RATE_IN_PACKETS	If set, gtf rate is configured in packets per second
BCM_SAT_GTF_RATE_WITH_GRANULARITY	If set, the granularity can be changed; if not, granularity will be 1

Section 6.11.2: SAT GTF Bandwidth APIs

APIs for Section 6.11.2: SAT GTF Bandwidth APIs

bcm_sat_gtf_bandwidth_t_init

Initialize a bcm_sat_gtf_bandwidth_t structure.

SYNOPSIS

```
#include <bcm/sat.h>
void bcm_sat_gtf_bandwidth_t_init(bcm_sat_gtf_bandwidth_t *bw);
```

PARAMETERS

bw	(OUT) Pointer to SAT gtf bandwidth structure to initialize

DESCRIPTION

Initializes a SAT gtf bandwidth structure to default values. This function must be used to initialize any SAT gtf bandwidth structure before passing it to an API function.

bcm_sat_gtf_bandwidth_set

Set gtf bandwidth.

SYNOPSIS

```
#include <bcm/sat.h>
int bcm_sat_gtf_bandwidth_set(
    int unit,
    bcm_sat_gtf_t gtf_id,
    int priority,
    bcm_sat_gtf_bandwidth_t *bw);
```

PARAMETERS

unit	(IN) BCM device number
gtf_id	(IN) GTF ID
priority	(IN) -1 indicates common; 0 indicates cir; 1 indicates pir
bw	(IN) Pointer to SAT bandwidth structure

DESCRIPTION

Set gtf bandwidth.

RETURNS

BCM_E_NONE	Operation completed successfully
BCM_E_NOT_FOUND	Attempt to set bandwidth for GTF which does not exist
BCM_E_PARAM	Attempt to set gtf bandwidth with invalid parameters
BCM_E_UNAVAIL	current API is not supported on the specified unit
BCM_E_TIMEOUT	Unable to obtain resource lock
BCM_E_INTERNAL	Unable to release resource lock / Failed to write memory

bcm_sat_gtf_bandwidth_get

Get gtf bandwidth.

SYNOPSIS

```
#include <bcm/sat.h>
int bcm_sat_gtf_bandwidth_get(
    int unit,
    bcm_sat_gtf_t gtf_id,
    int priority,
    bcm_sat_gtf_bandwidth_t *bw);
```

PARAMETERS

unit	(IN) BCM device number
gtf_id	(IN) GTF ID
priority	(IN) -1 indicates common; 0 indicates cir; 1 indicates pir

bw	(IN/OUT) Pointer to SAT bandwidth structure

DESCRIPTION

Get gtf bandwidth.

RETURNS

BCM_E_NONE	Operation completed successfully
BCM_E_NOT_FOUND	Attempt to get bandwidth for GTF which does not exist
BCM_E_PARAM	Attempt to get bandwidth with invalid parameters
BCM_E_UNAVAIL	current API is not supported on the specified unit
BCM_E_TIMEOUT	Unable to obtain resource lock
BCM_E_INTERNAL	Unable to release resource lock / Failed to write memory

Section 6.12: Statistics

```
/* This function is used to move the old set of counters to new free location provided by user. */
int stat_custom_counter_id_move(
    int unit,
    bcm_stat_custom_base_index_action_t idx_action,
    bcm_stat_custom_counter_info_t counter_info_old,
    bcm_stat_custom_counter_info_t counter_info_new
)


/* Initialize flex stat counter information */
void stat_custom_counter_info_t_init(
    bcm_stat_custom_counter_info_t counter_info
```

```
)


/* Associate an accounting object to customized group mode with user configured pool id. */
int stat_custom_group_id_create(
    int unit,
    uint32 mode_id,
    bcm_stat_object_t object,
    uint32 pool_id,
    uint32 base_idx,
    bcm_stat_custom_counter_info_t counter_info
)


typedef struct stat_custom_counter_info_s {
    uint32 ctr_tbl_mode_id; /* Counter table Mode Identifier */
    uint32 ctr_tbl_pool_id; /* Counter table Pool Id */
    uint32 ctr_tbl_pool_base_idx; /* Counter table Pool Base Index */
    uint32 offset_tbl_base_idx; /* Offset table Pool Base Index */
    uint32 num_counters; /* Number of counter entries created */
    uint32 stat_counter_id; /* Stat Counter Id */
    bcm_stat_object_t object; /* Stat Accounting object */
} stat_custom_counter_info_t;


/* bcm_custom_stat_trigger_e */
typedef enum bcm_custom_stat_trigger_e {
  ...
  DbgCntRxHigigLoopbackDrop  101  /* Rx: HiGig loopback packets dropped. */
  DbgCntRxHigigUnknownOpcodeDrop  102  /* Rx: HiGig packets dropped due to unknown
opcode. */
  DbgCntRxIpInIpDrop  103  /* Rx: IPinIP tunnel process failure. */
  DbgCntRxMimDrop  104  /* Rx: MiM tunnel process failure. */
  DbgCntRxTunnelInterfaceError  105  /* Rx: IP multicast tunnel incoming interface is incorrect. */
  DbgCntRxMplsControlWordInvalid  106  /* Rx: MPLS invalid control word. */
  DbgCntRxMplsGalNotBosDrop  107  /* Rx: MPLS Generic Label is not BOS. */
  DbgCntRxMplsPayloadInvalid  108  /* Rx: MPLS invalid payload. */
  DbgCntRxMplsEntropyDrop  109  /* Rx: MPLS entropy label in unallowed range. */
  DbgCntRxMplsLabelLookupMiss  110  /* Rx: MPLS label lookup miss. */
  DbgCntRxMplsActionInvalid  111  /* Rx: MPLS invalid action. */
  DbgCntRxTunnelTtlError  112  /* Rx: Tunnel TTL check failure. */
  DbgCntRxTunnelShimError  113  /* Rx: Tunnel shim header error. */
  DbgCntRxTunnelObjectValidationFail  114  /* Rx: Tunnel termination failure as not all packet
forwarding objects got assigned. */
  DbgCntRxVlanMismatch  115  /* Rx: VLAN ID invalid. */
  DbgCntRxVlanMemberMismatch  116  /* Rx: Ingress port not in VLAN membership. */
  DbgCntRxTpidMismatch  117  /* Rx: VLAN TPID mismatch. */
  DbgCntRxPrivateVlanMismatch  118  /* Rx: PVLAN VID mismatch. */
  DbgCntRxMacIpBindError  119  /* Rx: MAC to IP bind check failure. */
  DbgCntRxVlanTranslate2LookupMiss  120  /* Rx: VLAN Translate2 lookup miss. */
  DbgCntRxL3TunnelLookupMiss  121  /* Rx: L3 Tunnel lookup miss. */
  DbgCntRxMplsLookupMiss  122  /* Rx: MPLS lookup miss. */
```

```
  DbgCntRxVlanTranslate1LookupMiss  123  /* Rx: VLAN Translate1 lookup miss. */
  DbgCntRxFcoeVftDrop  124  /* Rx: FVT header error. */
  DbgCntRxFcoeSrcBindError  125  /* Rx: FCOE source bind check failure. */
  DbgCntRxFcoeSrcFpmaError  126  /* Rx: Source FPMA prefix check failure. */
  DbgCntRxVfiP2pDrop  127  /* Rx: PT2PT service but packets arrived from a VP that is not
programmed in PT2PT connection. */
  DbgCntRxStgDrop  128  /* Rx: Spanning tree state not in forwarding state. */
  DbgCntRxCompositeError  129  /* Rx: HW related error like lookup table with parity error. */
  DbgCntRxBpduDrop  130  /* Rx: BPUD packets dropped due to PORT_TABLE.DROP_BPDU. */
  DbgCntRxProtocolDrop  131  /* Rx: Protocol packets dropped due to
PROTOCOL_PKT_CONTROL config. */
  DbgCntRxUnknownMacSaDrop  132  /* Rx: Unknown source MAC packets dropped due to
CML_OFFSET.DROP. */
  DbgCntRxSrcRouteDrop  133  /* Rx: Packets dropped due to MACSA is multicast(bit 40 == 1). */
  DbgCntRxL2SrcDiscardDrop  134  /* Rx: L2 SRC_DISCARD drop. */
  DbgCntRxL2SrcStaticMoveDrop  135  /* Rx: Port movement of static L2 addr. */
  DbgCntRxL2DstDiscardDrop  136  /* Rx: L2 DST_DISCARD drop. */
  DbgCntRxL3DisableDrop  137  /* Rx: Packets dropped due to
PORT_TABLE.V4L3_ENABLE/V6L3_ENABLE unset. */
  DbgCntRxMacSaEqual0Drop  138  /* Rx: Packets dropped due to MACSA == 0. */
  DbgCntRxVlanCrossConnectOrNonUnicastDrop  139  /* Rx: L2 forwarding lookup miss or PBT
non unicast packets drop. */
  DbgCntRxTimeSyncDrop  140  /* Rx: Network time sync packets dropped. */
  DbgCntRxIpmcDrop  141  /* Rx: IPMC process failure. */
  DbgCntRxMyStationDrop  142  /* Rx: MY_STATION.DISCARD drop. */
  DbgCntRxPrivateVlanVpFilterDrop  143  /* Rx: Private vlan drop based on src/dst virtual port
type. */
  DbgCntRxL3DstDiscardDrop  144  /* Rx: L3 DST_DISCARD drop. */
  DbgCntRxTunnelDecapEcnError  145  /* Rx: Tunnel decap ECN error. */
  DbgCntRxL3SrcDiscardDrop  146  /* Rx: L3 SRC_DISCARD drop. */
  DbgCntRxFcoeZoneLookupMiss  147  /* Rx: FCOE_ZONE lookup miss. */
  DbgCntRxL3TtlError  148  /* Rx: L3 TTL check failure. */
  DbgCntRxL3HeaderError  149  /* Rx: L3 header error. */
  DbgCntRxL2HeaderError  150  /* Rx: L2 header error. */
  DbgCntRxL3DstLookupDrop  151  /* Rx: IPMC processing drop, or L3_DESTINATION == 0 for
L3UC packets. */
  DbgCntRxL2TtlError  152  /* Rx: L2 ZONE TTL check failure. */
  DbgCntRxL2RpfError  153  /* Rx: Incoming port/lag/svp check failure enforced L2 entry. */
  DbgCntRxTagUntagDiscardDrop  154  /* Rx: Packets dropped due to
PORT_DIS_UNTAG/PORT_DIS_TAG. */
  DbgCntRxStormControlDrop  155  /* Rx: Storm control metering drop. */
  DbgCntRxFcoeZoneError  156  /* Rx: FCOE_ZONE check failure. */
  DbgCntRxFieldChangeL2NoRedirectDrop  157  /* Rx: FP Changes L2 Fields packets is not
eligible for routing. Redirect action must be deployed to select packet destination. */
  DbgCntRxNextHopDrop  158  /* Rx: NEXT_HOP drop. */
  DbgCntRxNatDrop  159  /* Rx: NAT process failure, e.g. destination lookup miss or Hairpin drop.
*/
  DbgCntIngressProtectionDataDrop  160  /* Rx: Protection data drop. */
  DbgCntRxSrcKnockoutDrop  161  /* Rx: SGPP == DGPP or SVP == DVP, etc. */
  DbgCntRxMplsSeqNumberError  162  /* Rx: MPLS control word sequence number error. */
  DbgCntRxBlockMaskDrop  163  /* Rx: Packets dropped due to bitmap of ports to be blocked. */
  DbgCntRxDlbRhResolutionError  164  /* Rx: DLB or RH resolution error. */
  DbgCntTxFwdDomainNotFoundDrop  165  /* Tx: Forwarding domain not found. */
  DbgCntTxNotFwdDomainMemberDrop  166  /* Tx: Not forwarding domain member. */
```

```
  DbgCntTxIpmcL2SrcPortPruneDrop  167  /* Tx: IPMC L2 self port drop, VLAN ID in the packet is
the same as the VLAN ID in EGR_L3_INTF. */
  DbgCntTxNonUnicastPruneDrop  168  /* Tx: Non unicast pruning. */
  DbgCntTxSvpRemoveDrop  169  /* Tx: Virtual port pruning. */
  DbgCntTxVplagPruneDrop  170  /* Tx: VPLAG pruning. */
  DbgCntTxSplitHorizonDrop  171  /* Tx: Packets dropped due to split horizon check. This is done
by enabling pruning for DVP network group. */
  DbgCntTxMmuPurgeDrop  172  /* Tx: Packets dropped due to MMU purge. */
  DbgCntTxStgDisableDrop  173  /* Tx: Packets dropped due to STG disabled. */
  DbgCntTxEgressPortDrop  174  /* Tx: Packets dropped due to EGR_LPORT_PROFILE.DROP. */
  DbgCntTxEgressFilterDrop  175  /* Tx: Packets dropped by EFP. */
  DbgCntTxVisibilityDrop  176  /* Tx: Visibility packets dropped. */
  DbgCntNum  177  /* Must be last one. */


typedef enum stat_custom_base_index_action_e {
    StatCustomBaseIdxAlloc = 0,/* Counter Base Index Alloc */
    StatCustomBaseIdxMove = 1,/* Counter Base Index Move */
} stat_custom_base_index_action_t;
```

Section 6.13: Switch Control

```
#define BCM_SWITCH_OBM_PRIORITY_LOSSY_LOW   0        /* This obm_priority is
                                                        mapped to OBM buffer
                                                        partition Lossy Low */
#define BCM_SWITCH_OBM_PRIORITY_LOSSY_HIGH  1        /* This obm_priority is
                                                        mapped to OBM buffer
                                                        partition Lossy High */
#define BCM_SWITCH_OBM_PRIORITY_LOSSLESS0   2        /* This obm_priority is
                                                        mapped to OBM buffer
                                                        partition Lossless0 */
#define BCM_SWITCH_OBM_PRIORITY_LOSSLESS1   3        /* This obm_priority is
                                                        mapped to OBM buffer
                                                        partition Lossless1 */
#define BCM_SWITCH_OBM_PRIORITY_EXPRESS     4        /* This obm_priority is
                                                        mapped to OBM buffer
                                                        partition Express */


/* Initialize an OBM classifier structure. */
void switch_obm_classifier_t_init(
    bcm_switch_obm_classifier_t switch_obm_classifier
)


/* Set multiple mapping from code_point, Dest MAC and Ethertype to a obm_priority based on
```

```
OBM classifier type. */
int switch_obm_classifier_mapping_multi_set(
    int unit,
    bcm_gport_t gport,
    bcm_switch_obm_classifier_type_t switch_obm_classifier_type,
    int array_count,
    bcm_switch_obm_classifier_t switch_obm_classifier
)


/* Set the mapping from code_point, Dest MAC and Ethertype to a obm_priority based on OBM
classifier type. */
int switch_obm_classifier_mapping_set(
    int unit,
    bcm_gport_t gport,
    bcm_switch_obm_classifier_type_t switch_obm_classifier_type,
    bcm_switch_obm_classifier_t switch_obm_classifier
)


/* Get the mapping from code_point, Dest MAC and Ethertype to a obm_priority based on OBM
classifier type. */
int switch_obm_classifier_mapping_get(
    int unit,
    bcm_gport_t gport,
    bcm_switch_obm_classifier_type_t switch_obm_classifier_type,
    bcm_switch_obm_classifier_t switch_obm_classifier
)


/* Get multiple mapping from port, Dest MAC and Ethertype to a obm_priority based on OBM
classifier type. */
int switch_obm_classifier_mapping_multi_get(
    int unit,
    bcm_gport_t port,
    bcm_switch_obm_classifier_type_t switch_obm_classifier_type,
    int array_max,
    bcm_switch_obm_classifier_t switch_obm_classifier,
    int array_count
)


typedef struct switch_obm_classifier_s {
    uint32 flags; /* For future use */
    bcm_mac_t obm_destination_mac; /* Destination MAC address */
    bcm_mac_t obm_destination_mac_mask; /* Destination MAC address mask */
    uint16 obm_ethertype; /* Ethertype to be matched */
    uint16 obm_ethertype_mask; /* Mask used in matching Ethertype */
    bcm_switch_obm_code_point_t obm_code_point; /*  Oversubscription Buffer Manager
classifier's code point.This field if overloaded for providing DSCP, MPLS traffic class, ETAG priority
code point, VLAN priority code point or Higig2 traffic class depending on the type of classifier. This
will be mapped to obm_priority value. */
    bcm_switch_obm_priority_t obm_priority; /* Oversubscription Buffer Manager classifier's priority
for given obm_code_point. */
```

```
} switch_obm_classifier_t;


typedef enum switch_obm_classifier_type_e {
    SwitchObmClassifierDscp = 0,/* OBM classifier of type DSCP */
    SwitchObmClassifierMpls = 1,/* OBM classifier of type MPLS */
    SwitchObmClassifierEtag = 2,/* OBM classifier of type ETAG */
    SwitchObmClassifierVlan = 3,/* OBM classifier of type VLAN */
    SwitchObmClassifierHigig2 = 4,/* OBM classifier of type Higig2 */
    SwitchObmClassifierField = 5,/* OBM classifier of type FIELD */
} switch_obm_classifier_type_t;



/* Switch controls. */
typedef enum bcm_switch_control_e {
  ..
  SwitchEtherDscpRemark  1121  /* Enable copying IP header.TOS to In-DSCP-EXP. */
  Switch__Count  1122  /* Must be last.  Not a usable value. */
} bcm_switch_control_t;



/* packet trace drop reason enums */
typedef enum bcm_switch_pkt_trace_drop_reason_e {
    bcmSwitchPktTraceDropReasonTunnelObjectValidationFail = 37,
                                /* Packet dropped due to Tunnel
                                   termination can't be done as not all
                                   packet forwarding objects got
                                   assigned. */
    bcmSwitchPktTraceDropReasonTunnelShimHeaderError = 38,
                                /* Packet dropped due to tunnel shim
                                   header error. */
    bcmSwitchPktTraceDropReasonTunnelTTLError = 39, /* Packet dropped due to tunnel TTL
                                   check fail. */
    bcmSwitchPktTraceDropReasonTunnelInterfaceCheckFail = 40, /* Packet dropped due to tunnel
                                   interface check failure. */
    bcmSwitchPktTraceDropReasonTunnelError = 41, /* Packet dropped due to tunnel error. */
    bcmSwitchPktTraceDropReasonTunnelAdapt1LookupMiss = 42, /* Packcet dropped due to
tunnel
                                   adaptation table 1 lookup miss. */
    bcmSwitchPktTraceDropReasonTunnelAdapt2LookupMiss = 43, /* Packcet dropped due to
tunnel
                                   adaptation table 2 lookup miss. */
    bcmSwitchPktTraceDropReasonTunnelAdapt3LookupMiss = 44, /* Packcet dropped due to
tunnel
                                   adaptation table 3 lookup miss. */
    bcmSwitchPktTraceDropReasonTunnelAdapt4LookupMiss = 45, /* Packcet dropped due to
tunnel
                                   adaptation table 4 lookup miss. */
    bcmSwitchPktTraceDropReasonCount = 46
} bcm_switch_pkt_trace_drop_reason_t;
```

Section 6.14: Time Configuration

bcm_time_bs_time_get

Retrieves broadsync time from device

SYNOPSIS

```
#include <bcm/time.h>
extern int bcm_time_bs_time_get( int unit, bcm_time_spec_t *bs_time);
```

PARAMETERS

unit	BCM device number
bs_time	(OUT) A bcm_time_spec_t structure to fill

DESCRIPTION

Retrieves broadsync time in seconds and nanoseconds format from the device. If the interface was configured in INPUT mode i.e. Broadsync slave the API will return the received time from the broadsync timecode signal. If the interface was configured as Broadsync master the API will return the output time for the current heartbeat.

RETURNS

BCM_E_NONE	- Success
BCM_E_UNAVAIL	- Feature is not available
BCM_E_XXX	- Any other error

Section 6.15: Packet Transmit and Receive

Table: bcm_pkt_t Structure Description

Field	*Type*	*Description*
egress_to_cpu_hdr_size	uint16	Size of egress to CPU header in CMIC devices. This header is applicable only in the new generation of CMIC called CMIC-X.

Section 6.16: Time-Sensitive Networking (TSN)

```
/* TSN priority map id */
typedef int bcm_tsn_pri_map_t;

/* Get various TAF configurations at specific gate */
int tsn_taf_control_get(
    int unit,
    int taf_gate,
    bcm_tsn_taf_control_t type,
    uint32 arg
)

/* set various TAF configurations at specific gate */
int tsn_taf_control_set(
    int unit,
    int taf_gate,
    bcm_tsn_taf_control_t type,
    uint32 arg
)

/* Create TAF Cos mapping profile */
int tsn_taf_cosq_mapping_profile_create(
    int unit,
    int cosq_profile
)

/* Destroy TAF Cos mapping profile */
int tsn_taf_cosq_mapping_profile_destroy(
    int unit,
    int cosq_profile
)
```

```
/* Get TAF Cos mapping profile */
int tsn_taf_cosq_mapping_profile_get(
    int unit,
    int cosq_profile,
    bcm_cos_t priority,
    bcm_cos_queue_t cosq
)

/* Set TAF Cos mapping profile */
int tsn_taf_cosq_mapping_profile_set(
    int unit,
    int cosq_profile,
    bcm_cos_t priority,
    bcm_cos_queue_t cosq
)

/* Traverse TAF Cos mapping profile */
int tsn_taf_cosq_mapping_profile_traverse(
    int unit,
    bcm_tsn_taf_cosq_mapping_profile_traverse_cb cb,
    void user_data
)

/* Register a callback for handling Tsn Taf events */
int tsn_taf_event_register(
    int unit,
    bcm_tsn_taf_event_types_t event_types,
    int taf_gate,
    bcm_tsn_taf_event_cb cb,
    void user_data
)

/* Unregister a callback for handling Tsn Taf events */
int tsn_taf_event_unregister(
    int unit,
    bcm_tsn_taf_event_types_t event_types,
    int taf_gate,
    bcm_tsn_taf_event_cb cb
)

/* Create a TAF gate */
int tsn_taf_gate_create(
    int unit,
    int flags,
    int taf_gate_id
)

/* Destroy a TAF gate */
int tsn_taf_gate_destroy(
    int unit,
    int taf_gate_id
)

/* Create profile of maximum bytes that pass through the TAF gate */
```

```
int tsn_taf_gate_max_bytes_profile_create(
    int unit,
    int taf_gate_id,
    uint64 max_bytes,
    int profile_id
)

/* Destroy profile of maximum bytes that pass through the TAF gate */
int tsn_taf_gate_max_bytes_profile_destroy(
    int unit,
    int taf_gate_id,
    int profile_id
)

/* Get profile of maximum bytes that pass through the TAF gate */
int tsn_taf_gate_max_bytes_profile_get(
    int unit,
    int taf_gate_id,
    int profile_id,
    uint64 max_bytes
)

/* Set profile of maximum bytes that pass through the TAF gate */
int tsn_taf_gate_max_bytes_profile_set(
    int unit,
    int taf_gate_id,
    int profile_id,
    uint64 max_bytes
)

/* Traverse profile of maximum bytes that pass through the TAF gate */
int tsn_taf_gate_max_bytes_profile_traverse(
    int unit,
    int taf_gate_id,
    bcm_tsn_taf_gate_max_bytes_profile_traverse_cb cb,
    void user_data
)

/* Get 64-bit counter value for specified TAF gate statistic type. */
int tsn_taf_gate_stat_get(
    int unit,
    int taf_gate_id,
    bcm_tsn_taf_gate_stat_t stat,
    uint64 val
)

/* Get lower 32-bit counter value for specified TAF gate statistic type. */
int tsn_taf_gate_stat_get32(
    int unit,
    int taf_gate_id,
    bcm_tsn_taf_gate_stat_t stat,
    uint32 val
)
```

```
/* Get 64-bit counter value for multiple TAF gate statistic types. */
int tsn_taf_gate_stat_multi_get(
    int unit,
    int taf_gate_id,
    int nstat,
    bcm_tsn_taf_gate_stat_t stat_arr,
    uint64 val_arr
)

/* Get lower 32-bit counter value for multiple TAF gate statistic types. */
int tsn_taf_gate_stat_multi_get32(
    int unit,
    int taf_gate_id,
    int nstat,
    bcm_tsn_taf_gate_stat_t stat_arr,
    uint32 val_arr
)

/* Set 64-bit counter value for multiple TAF gate statistic types. */
int tsn_taf_gate_stat_multi_set(
    int unit,
    int taf_gate_id,
    int nstat,
    bcm_tsn_taf_gate_stat_t stat_arr,
    uint64 val_arr
)

/* Set lower 32-bit counter value for multiple TAF gate statistic types. */
int tsn_taf_gate_stat_multi_set32(
    int unit,
    int taf_gate_id,
    int nstat,
    bcm_tsn_taf_gate_stat_t stat_arr,
    uint32 val_arr
)

/* Set 64-bit counter value for specified TAF gate statistic type */
int tsn_taf_gate_stat_set(
    int unit,
    int taf_gate_id,
    bcm_tsn_taf_gate_stat_t stat,
    uint64 val
)

/* Set lower 32-bit counter value for specified TAF gate statistic type. */
int tsn_taf_gate_stat_set32(
    int unit,
    int taf_gate_id,
    bcm_tsn_taf_gate_stat_t stat,
    uint32 val
)

/* Get the specified hardware statistics (64-bit) from the device. */
int tsn_taf_gate_stat_sync_get(
```

```
    int unit,
    int taf_gate_id,
    bcm_tsn_taf_gate_stat_t stat,
    uint64 val
)

/* Get the specified hardware statistics (32-bit) from the device. */
int tsn_taf_gate_stat_sync_get32(
    int unit,
    int taf_gate_id,
    bcm_tsn_taf_gate_stat_t stat,
    uint32 val
)

/* Get multiple hardware statistics (64-bit) from the device. */
int tsn_taf_gate_stat_sync_multi_get(
    int unit,
    int taf_gate_id,
    int nstat,
    bcm_tsn_taf_gate_stat_t stat_arr,
    uint64 val_arr
)

/* Get multiple hardware statistics (64-bit) from the device. */
int tsn_taf_gate_stat_sync_multi_get32(
    int unit,
    int taf_gate_id,
    int nstat,
    bcm_tsn_taf_gate_stat_t stat_arr,
    uint32 val_arr
)

/* Traverse TAF gate */
int tsn_taf_gate_traverse(
    int unit,
    bcm_tsn_taf_gate_traverse_cb cb,
    void user_data
)

/* Commit TAF profile. */
int tsn_taf_profile_commit(
    int unit,
    int taf_gate,
    bcm_tsn_taf_profile_id_t pid
)

/* Create TAF profile. */
int tsn_taf_profile_create(
    int unit,
    int taf_gate,
    bcm_tsn_taf_profile_t profile,
    bcm_tsn_taf_profile_id_t pid
)
```

```
/* Destroy TAF profile. */
int tsn_taf_profile_destroy(
    int unit,
    int taf_gate,
    bcm_tsn_taf_profile_id_t pid
)

/* Destroy all TAF profiles. */
int tsn_taf_profile_destroy_all(
    int unit,
    int taf_gate
)

/* Get TAF profile. */
int tsn_taf_profile_get(
    int unit,
    int taf_gate,
    bcm_tsn_taf_profile_id_t pid,
    bcm_tsn_taf_profile_t profile
)

/* Set TAF profile. */
int tsn_taf_profile_set(
    int unit,
    int taf_gate,
    bcm_tsn_taf_profile_id_t pid,
    bcm_tsn_taf_profile_t profile
)

/* Get TAF profile status. */
int tsn_taf_profile_status_get(
    int unit,
    int taf_gate,
    bcm_tsn_taf_profile_id_t pid,
    bcm_tsn_taf_profile_status_t status
)

/* Initialize the bcm_tsn_taf_profile_status_t structure. Clear the content to zero */
void tsn_taf_profile_status_t_init(
    bcm_tsn_taf_profile_status_t profile_status
)

/* Initialize the bcm_tsn_taf_profile_t structure. Clear the content to zero. */
void tsn_taf_profile_t_init(
    bcm_tsn_taf_profile_t profile
)

/* Traverse configured TAF profile */
int tsn_taf_profile_traverse(
    int unit,
    int taf_gate,
    bcm_tsn_taf_profile_traverse_cb cb,
    void user_data
)
```

```
/* Get the TAF gate status */
int tsn_taf_status_get(
    int unit,
    int taf_gate,
    bcm_tsn_taf_status_t type,
    uint32 arg
)

typedef struct tsn_taf_entry_s {
    uint32 ticks; /* The scheduling interval in ticks */
    uint32 state; /* Specify gate state (1:open, 0:close)  */
    int cos_profile; /* Class of service (COS) mapping profile, value 0 for default */
    int maxbyte_profile; /* Max bytes profile, value 0 for default */
} tsn_taf_entry_t;

typedef struct tsn_taf_event_types_s {
    SHR_BITDCL w;
} tsn_taf_event_types_t;

typedef struct tsn_taf_profile_s {
    int num_entries; /* Specify the number of entries in the calendar table. */
    bcm_tsn_taf_entry_t entries; /* Specify the interval, gate states, attribute entry sets in the
calendar table. */
    bcm_ptp_timestamp_t ptp_base_time; /* Specify base time in PTP time. */
    uint32 ptp_cycle_time; /* Specify cycle time in ns. */
    uint32 ptp_max_cycle_extension; /* Specify max cycle extension time in ns. Maximum time
allowed for a cycle to be extended before profile change. */
} tsn_taf_profile_t;

typedef struct tsn_taf_profile_status_s {
    bcm_tsn_taf_profile_state_t profile_state; /* profile state */
    bcm_ptp_timestamp_t config_change_time; /* The actual config change time */
    int num_entries; /* The actual number of entries in hardware calendar table. */
    bcm_tsn_taf_entry_t entries; /* The actual entries in hardware calendar table. */
} tsn_taf_profile_status_t;

typedef enum tsn_taf_control_e {
    TsnTafControlEnable = 0,/* Enable the TAF. */
    TsnTafControlUsePTPTime = 1,/* PTP time is used for TAF synchronous operation. */
    TsnTafControlInitGateState = 2,/* Gate state when TAF is disabled. Or the init gate state in the
*Restart* process. */
    TsnTafControlInitCosProfile = 3,/* Cos Profile when TAF is disabled. Or the init cos Profile in the
*Restart* process. */
    TsnTafControlInitMaxBytesProfile = 4,/* Max bytes Profile when TAF is disabled. Or the init max
bytes Profile in the *Restart* process. */
    TsnTafControlErrGateState = 5,/* Gate state when error condition occurs during an old cycle is
running. */
    TsnTafControlErrCosProfile = 6,/* Cos profile when error condition occurs during an old cycle is
running. */
    TsnTafControlErrMaxByteProfile = 7,/* Max bytes profile when error condition occurs during an
old cycle is running. */
    TsnTafControlGateCloseControl = 8,/* if set, it will override the gate purge and drop decisions
during gate close state and it will let the packet go. */
```

```
    TsnTafControlBytesLeftCheckEnable = 9,/* Enable byte left check or not */
    TsnTafControlGateControlTickInterval = 10,/* Gate control ticks interval in ns. */
    TsnTafControlTAFPTPLock = 11,/* Indicate the PTP time is locked or out of sync. This type is
on system basis. */
    TsnTafControlCount = 12,/* This should be the last one. */
} tsn_taf_control_t;

typedef enum tsn_taf_event_type_e {
    TsnTafEventTAFProfileChange = 0,/* TAF: the profile change is completed. */
    TsnTafEventTAFECCErr = 1,/* TAF: uncorrectable ECC error in profile calendar table. */
    TsnTafEventTAFNextCycleTimeErr = 2,/* TAF: next cycle start time has passed. */
    TsnTafEventTAFBaseTimeErr = 3,/* TAF: base time for the new profile has passed.  */
    TsnTafEventTAFProfilePtrErr = 4,/* TAF: index to the profile calendar table is overrun. */
    TsnTafEventTAFPTPOutOfSyncErr = 5,/* TAF: IEEE1588 PTP time is out of sync. */
    TsnTafEventCount = 6,/* This should be the last one. */
} tsn_taf_event_type_t;

typedef enum tsn_taf_gate_stat_e {
    TsnTafGateStatPassed = 0,/* Number of packets passed TAF gate */
    TsnTafGateStatDrop = 1,/* Number of packets dropped by TAF gate */
    TsnTafGateStatCount = 2,/* Always last. Not a usable value. */
} tsn_taf_gate_stat_t;

typedef enum tsn_taf_profile_state_e {
    TsnTafProfileInit = 0,/* the state after profile created. */
    TsnTafProfileCommitted = 1,/* the state after profile committed but not yet written to HW table */
    TsnTafProfilePending = 2,/*  the state after profile is set to hardware but not yet effective */
    TsnTafProfileActive = 3,/* the state indicate the profile is effective */
    TsnTafProfileExpired = 4,/* The profile is no longer effective. */
    TsnTafProfileError = 5,/* the state indicate the profile is error when any error event happens */
    TsnTafProfileCount = 6,/* This should be the last one. */
} tsn_taf_profile_state_t;

typedef enum tsn_taf_status_e {
    TsnTafStatusGateState = 0,/* The current TAF gate state. */
    TsnTafStatusGateStateSet = 1,/* Reflect which control gate source is selected. The value = 0
indicates the gate state is from INIT GATE STATE settings, value = 1 indicates the gate state is
from active control list table, value = 2 indicates the gate state is from ERR GATE STATE settings
*/
    TsnTafStatusGateCosProfile = 2,/* The Cos profile this TAF gate use. */
    TsnTafStatusGateMaxBytesProfile = 3,/* The max bytes profile this TAF gate use. */
    TsnTafStatusTickGranularity = 4,/* The granularity of the calendar table time clock in ns. */
    TsnTafStatusMaxCalendarEntries = 5,/* The max entries could be support in calendar table. */
    TsnTafStatusGatMaxBytesLeft = 6,/* Bytes available that can pass through that gate during
current open state. */
    TsnTafStatusCount = 7,/* This should be the last one. */
} tsn_taf_status_t;



/* TSN priority mapping entry (one instance for one priority) */

typedef struct bcm_tsn_pri_map_entry_s {
```

```
    uint32 flags;          /* See BCM_TSN_PRI_MAP_ENTRY_xxx flags */
    int flow_offset;       /* Flow ID offset (0~7) */
    int new_int_pri;       /* New internal priority */
    int profile_id;        /* Profile ID */
    int taf_gate_express;  /* TAF gate for express traffic */
    int taf_gate_preempt;  /* TAF gate for preemptable traffic */
} bcm_tsn_pri_map_entry_t;



/* TSN controls */
typedef enum bcm_tsn_control_e {
    bcmTsnControlSrAgeTickInterval = 0, /* Age timer tick interval in msecs */
    bcmTsnControlSrAgeOutEnable = 1,    /* Global age out enable */
    bcmTsnControlSrSeqNumWindowResetMode = 2, /* See

BCM_TSN_CONTROL_SR_SEQNUM_WINDOW_RESET_MODE_xxx */
    bcmTsnControlSrLooseOutOfOrder = 3, /* Loose out of order definition. An SR
                                           packet is considered out of order if
                                           it's in the acceptance window and the
                                           equence number is less than previous
                                           received packet. */
    bcmTsnControlSrHsrEthertype = 4,    /* The Ethertype value for the HSR tag */
    bcmTsnControlSrPrpEthertype = 5,    /* The Ethertype (suffix) value for the
                                           PRP RCT trailer */
    bcmTsnControlSrDot1cbEthertype = 6, /* The Ethertype value for the 802.1CB
                                           tag */
    bcmTsnControlIngressMtuProfile = 7, /* Ingress MTU profile ID (per port) */
    bcmTsnControlEgressMtuPriorityMap = 8, /* Priority mapping for egress MTU check
                                           (per-port) */
```

```
    bcmTsnControlIngressStuProfile = 9, /* Ingress STU profile ID (per port) */
    bcmTsnControlEgressStuPriorityMap = 10, /* Priority mapping for egress STU check
                                (per-port) */
    bcmTsnControlTafEnable = 11,        /* Enable TAF (Time Aware Filtering)
                                (per-port) */
    bcmTsnControlTafGatePriMap = 12,    /*  TAF gate priority mapping (per-port) */
    bcmTsnControlTafGatePriMapL2Select = 13, /* TAF gate priority mapping lookup by
                                MAC-SA(0) or MAC-DA(1) (per-port) */
    bcmTsnControlCount = 14             /* Always last. Not a usable value. */
} bcm_tsn_control_t;



/* TSN priority map type */
typedef enum bcm_tsn_pri_map_type_e {
    bcmTsnPriMapTypeTsnFlow = 0,        /* Mapping priority to TSN flows (TSN
                                circuit ID) */
    bcmTsnPriMapTypeSrFlow = 1,         /* Mapping priority to SR flows */
    bcmTsnPriMapTypeEgressMtuProfile = 2, /* Mapping priority to egress MTU
                                profile index */
    bcmTsnPriMapTypeEgressStuProfile = 3, /* Mapping priority to egress STU
                                profile index */
    bcmTsnPriMapTypeTafGate = 4,        /* Mapping priority to TAF gate */
    bcmTsnPriMapTypeCount = 5           /* Always last. Not a usable value. */
} bcm_tsn_pri_map_type_t;

/* TSN stat types */
typedef enum bcm_tsn_stat_e {
    bcmTsnStatIngressSrTaggedPackets = 0, /* Number of SR tagged packets received */
```

```
bcmTsnStatIngressNonLinkLocalPackets = 1, /* Number of all non-link-local packets
                                             received */
bcmTsnStatIngressSrWrongLanPackets = 2, /* Number of PRP packets with wrong LAN
                                           ID */
bcmTsnStatIngressSrRxErrorPackets = 3, /* Number of packets received with
                                          invalid flow or dropped due to errors */
bcmTsnStatIngressSrTagErrorPackets = 4, /* Number of packets received with
                                           invalid SR tag */
bcmTsnStatIngressSrDuplicatePackets = 5, /* Number of duplicate SR packets
                                            received */
bcmTsnStatIngressSrOutOfOrderSrPackets = 6, /* Number of SR packets received out of
                                               order */
bcmTsnStatIngressSrOwnRxPackets = 7, /* Number of SR packets received that
                                        originated from this device */
bcmTsnStatIngressSrUnexpectedPackets = 8, /* Number of packets received with wrong
                                             SR tag */
bcmTsnStatIngressMtuErrorPackets = 9, /* Number of packets received that have
                                         violated MTU size */
bcmTsnStatIngressStuErrorPackets = 10, /* Number of packets received that have
                                          violated STU size */
bcmTsnStatIngressSrAcceptedSrPackets = 11, /* Number of SR packets that are
                                              accepted and pass the duplicate
                                              discard mechanism */
bcmTsnStatIngressSrSaSrcFilteredPackets = 12, /* Number of SR packets that dropped due
                                                 to source port filtering */
bcmTsnStatIngressTafMeterDrop = 13, /* Number of packets got dropped by the
                                       meter */
bcmTsnStatIngressTafGatePass = 14,  /* Number of packets passed through the
```

```
                                TAF gate */
    bcmTsnStatIngressTafGateCloseDrop = 15, /* Number of packets got dropped by TAF
                                gate during close state */
    bcmTsnStatIngressTafGateNoByteDrop = 16, /* Number of packets got dropped by TAF
                                gate during open state because of max
                                bytes check failed */
    bcmTsnStatIngressTafMtuPass = 17,   /* Number of packets passed the MTU
                                check */
    bcmTsnStatEgressSrTaggedPackets = 18, /* Number of SR tagged packets
                                transmitted */
    bcmTsnStatEgressNonLinkLocalPackets = 19, /* Number of all non-link-local packets
                                transmitted */
    bcmTsnStatEgressMtuErrorPackets = 20, /* Number of packets transmitted that
                                have violated MTU size */
    bcmTsnStatEgressStuErrorPackets = 21, /* Number of packets transmitted that
                                have violated STU size */
    bcmTsnStatCount = 22            /* Always last. Not a usable value. */
} bcm_tsn_stat_t;
```

Section 6.17: BCM types

```
typedef struct vlan_action_set_s {
    bcm_vlan_t new_outer_vlan; /* New outer VLAN for Add/Replace actions. */
    bcm_vlan_t new_inner_vlan; /* New inner VLAN for Add/Replace actions. */
    uint8 new_inner_pkt_prio; /* New inner packet priority for Add/Replace actions. */
    uint8 new_outer_cfi; /* New outer packet CFI for Add/Replace actions. */
    uint8 new_inner_cfi; /* New inner packet CFI for Add/Replace actions. */
    bcm_if_t ingress_if; /* L3 Ingress Interface. */
```

```
    int priority; /* Internal or packet priority. */
    bcm_vlan_action_t dt_outer; /* Outer-tag action for double-tagged packets. */
    bcm_vlan_action_t dt_outer_prio; /* Outer-priority-tag action for double-tagged packets. */
    bcm_vlan_action_t dt_outer_pkt_prio; /* Outer packet priority action for double-tagged packets.
*/
    bcm_vlan_action_t dt_outer_cfi; /* Outer packet CFI action for double-tagged packets. */
    bcm_vlan_action_t dt_inner; /* Inner-tag action for double-tagged packets. */
    bcm_vlan_action_t dt_inner_prio; /* Inner-priority-tag action for double-tagged packets. */
    bcm_vlan_action_t dt_inner_pkt_prio; /* Inner packet priority action for double-tagged packets. */
    bcm_vlan_action_t dt_inner_cfi; /* Inner packet CFI action for double-tagged packets. */
    bcm_vlan_action_t ot_outer; /* Outer-tag action for single-outer-tagged packets. */
    bcm_vlan_action_t ot_outer_prio; /* Outer-priority-tag action for single-outer-tagged packets. */
    bcm_vlan_action_t ot_outer_pkt_prio; /* Outer packet priority action for single-outer-tagged
packets. */
    bcm_vlan_action_t ot_outer_cfi; /* Outer packet CFI action for single-outer-tagged packets. */
    bcm_vlan_action_t ot_inner; /* Inner-tag action for single-outer-tagged packets. */
    bcm_vlan_action_t ot_inner_pkt_prio; /* Inner packet priority action for single-outer-tagged
packets. */
    bcm_vlan_action_t ot_inner_cfi; /* Inner packet CFI action for single-outer-tagged packets. */
    bcm_vlan_action_t it_outer; /* Outer-tag action for single-inner-tagged packets. */
    bcm_vlan_action_t it_outer_pkt_prio; /* Outer packet priority action for single-inner-tagged
packets. */
    bcm_vlan_action_t it_outer_cfi; /* Outer packet CFI action for single-inner-tagged packets. */
    bcm_vlan_action_t it_inner; /* Inner-tag action for single-inner-tagged packets. */
    bcm_vlan_action_t it_inner_prio; /* Inner-priority-tag action for single-inner-tagged packets. */
    bcm_vlan_action_t it_inner_pkt_prio; /* Inner packet priority action for single-inner-tagged
packets. */
    bcm_vlan_action_t it_inner_cfi; /* Inner packet CFI action for single-inner-tagged packets. */
    bcm_vlan_action_t ut_outer; /* Outer-tag action for untagged packets. */
    bcm_vlan_action_t ut_outer_pkt_prio; /* Outer packet priority action for untagged packets. */
    bcm_vlan_action_t ut_outer_cfi; /* Outer packet CFI action for untagged packets. */
    bcm_vlan_action_t ut_inner; /* Inner-tag action for untagged packets. */
    bcm_vlan_action_t ut_inner_pkt_prio; /* Inner packet priority action for untagged packets. */
    bcm_vlan_action_t ut_inner_cfi; /* Inner packet CFI action for untagged packets. */
    bcm_vlan_pcp_action_t outer_pcp; /* Outer tag's pcp field action of outgoing packets. */
    bcm_vlan_pcp_action_t inner_pcp; /* Inner tag's pcp field action of outgoing packets. */
    bcm_policer_t policer_id; /* Base policer to be used */
    uint16 outer_tpid; /* New outer-tag's tpid field for modify action */
    uint16 inner_tpid; /* New inner-tag's tpid field for modify action */
    bcm_vlan_tpid_action_t outer_tpid_action; /* Action of outer-tag's tpid field */
    bcm_vlan_tpid_action_t inner_tpid_action; /* Action of inner-tag's tpid field */
    int action_id; /* Action Set index */
    uint32 class_id; /* Class ID */
    bcm_tsn_pri_map_t taf_gate_primap; /* TAF (Time Aware Filtering) gate priority mapping */
    uint32 flags; /* BCM_VLAN_ACTION_SET_xxx. */
} vlan_action_set_t;
```

Section 6.18: UDF Resources Management

Table: UDF Abstract Packet Format Types enumeration for 'bcm_udf_abstract_pkt_format_t'.

bcmUdfAbstractPktFormatXxx	*Description*
UdfAbstractPktFormatTcpUnknownL5WithIpExtnHdr	Abstract from the start of first byte of TCP header for unknown L5 packet.

Section 6.19: VLAN Management

```
/* Change to vlan_ctrl_port */
typedef enum {
    bcmVlanPortPreferIP4,
    bcmVlanPortPreferMac,
    bcmVlanTranslateIngressEnable,
    bcmVlanTranslateIngressHitDrop,
    bcmVlanTranslateIngressMissDrop,
    bcmVlanTranslateEgressEnable,
    bcmVlanTranslateEgressMissDrop,
    bcmVlanTranslateEgressMissUntaggedDrop,
    bcmVlanTranslateEgressMissUntag,
    bcmVlanLookupMACEnable,
    bcmVlanLookupIPEnable,
    bcmVlanPortUseInnerPri,
    bcmVlanPortVerifyOuterTpid,
    bcmVlanPortOuterTpidSelect,
    bcmVlanPortIgnorePktTag,
    bcmVlanPortUntaggedDrop,
    bcmVlanPortPriTaggedDrop,
    bcmVlanPortDoubleLookupEnable,
    bcmVlanPortLookupTunnelEnable,
    bcmVlanPortIgnoreInnerPktTag,
    bcmVlanPortJoinAllVlan,
    bcmVlanPortOamUseXlatedInnerVlan,
    bcmVlanPortTranslateFcoeKeyFirst,
    bcmVlanPortTranslateFcoeKeySecond,
    bcmVlanPortTranslateEgressKey,
    bcmVlanPortTranslateEgressKeyFirst,
    bcmVlanPortTranslateEgressKeySecond
} bcm_vlan_control_port_t;
```

```
#define BCM_VLAN_FLAGS2_IVL             0x00000004 /* Configure VSI to use
                                        IVL key format
                                        (vid,vsi,mac) */
```

Section 6.20: VXLAN Management

```
typedef struct vxlan_vpn_config_s {
    uint32 flags; /* BCM_VXLAN_VPN_xxx. */
    bcm_vpn_t vpn; /* VXLAN VPN */
    uint32 vnid; /* VNID */
    uint8 pkt_pri; /* Packet Priority */
    uint8 pkt_cfi; /* Packet CFI */
    uint16 egress_service_tpid; /* Service TPID */
    bcm_vlan_t egress_service_vlan; /* Service VLAN */
    bcm_multicast_t broadcast_group;
    bcm_multicast_t unknown_unicast_group;
    bcm_multicast_t unknown_multicast_group;
    bcm_vlan_protocol_packet_ctrl_t protocol_pkt;
    bcm_vlan_t vlan; /* Outer VLAN */
    bcm_gport_t match_port_class; /* local port vlan domain */
    bcm_vlan_t default_vlan; /* default vlan for untag payload */
} vxlan_vpn_config_t;
```

Section 7: Test Statistics

Section 7.1: How to read the data

In cases where tables are shown below, the tables represent a spread of data gathered per device, per suite, and per release. The percentages represent the aggregate rate of failure for that suite when run against all variants of the family of devices. This data does not include results from DNX device regressions.

The below data is not meant to be a precise indication of quality but instead serves as a guideline for improvements release-over-release. Additionally, although some cells show 0% failures, this does not necessarily mean the feature is supported in the device - tests are run to validate the appropriate SDK support even for unsupported features on older devices to ensure graceful handling of all APIs. Finally, some devices have fewer columns listed if they were introduced recently.

Section 7.2: Overview

Each suite listed below is indicative of a specific module. Golden refers to a suite of tests that takes representation across multiple modules and serves as a sanity regression. Each suite contains tests of various types, loosely categorized as follows:

Test Categories	***Description***
Configuration Tests	Tests that verify that each API functions appropriately and can configure the device as expected.
Functionality Tests	Tests that further validate each of the API through functional use often requiring traffic to be run through the system.
Semantic Tests	Tests that ensure that the proper error handling mechanisms are working and users cannot crash the device through the API.

Section 7.2.1: Linux kernel versions used in this release

In SDK 6.5.12, the following Linux kernel versions were used in our main development and regression cycles with these external CPUs:

BCM9COMX2XMC86 (XLR): 4.4.6
BCM958712D4XMC (SLK): 3.14.65
BCM9X86D1508COME6 (GTS): 4.4.6

Please refer to the Broadcom Network Switching Software Platform Guide for more details about these CPUs. In this release, XLR was used in limited testing while BCM9XLP208XMC (WRX), BCM958525XMC (RSX), and BCM98548PPCXMC (GTO) are no longer part of our main focus for regression activity.

Testing using ARM-based CPU subsystem integrated processors (IPROC) was performed using kernel 4.4.39.

Section 7.3: Total Tests

The data below represents the number of unique cases for each release. The goal is to increase test coverage release over release but there may be instances where tests are consolidated which may yield a net reduction from one version to the next. Note that although a particular test case will execute for each and every chip, it is only counted once.

	sdk-6.5.12	***sdk-6.5.11***	***sdk-6.5.10***
golden	153	153	153
warmboot	5616	5544	4484
auth	17	17	17
bfd	123	123	123
bhh	159	159	159
chip	9	9	9
coe	663	663	663
cosq	813	813	810
custom	7	7	7
ea	108	108	108
eav	19	19	19
extender	49	49	49
fabric	7	7	7
failover	10	10	10

fcoe	37	37	37
field	1776	1771	1753
higigproxy	129	129	129
infra	114	114	114
ipfix	17	17	17
ipmc	132	130	125
l2	346	346	346
l2gre	33	33	33
l3	578	573	565
l3.alpm	560	550	550
link	27	26	26
mim	55	55	52
mirror	275	259	192
misc	24	24	24
mpls	588	586	571
multicast	29	29	29
niv	71	71	71
oam	400	400	400
pkt	59	56	55
port	493	492	490
proxy	48	48	45
ptp	124	124	118
qos	64	64	64
rate	21	21	21
rtag7	80	80	62
rx	58	58	51
ser	272	254	248
stack	119	119	119
stat	440	440	438
stg	42	42	42
switch	226	226	219

time	33	33	33
tlvMsg	13	13	13
trill	48	48	48
trunk	252	252	252
tunnel	133	133	133
subport	31	31	31
vlan	248	248	248
vxlan	321	319	225
wlan	17	17	17
Test Suite Total	16086	15949	14624

Section 7.4: API Test Results

In this release, all tested devices passed our DVAPI regressions with over 99.8% passing rate.

Section 7.5: Security Vulnerability Test Results

Starting in release SDK 6.5.10 we are reporting product security test results. These are scaling and semantic testing which verify that we properly handle errors and scaling to the limits. The table below shows the failing rate on the security suite.

	Total Tests	***% Pass***
minigolden	3	100%
warmboot	308	100%
cosq	270	100%
e2ecc	5	100%
ea	6	100%
eav	16	100%
fabric	4	100%
fcoe	3	100%
field	23	100%

higigproxy	43	100%
l2	134	100%
l3	26	100%
l3.alpm	216	100%
linkphy	7	100%
mim	1	100%
mirror	38	100%
mpls	25	100%
multicast	2	100%
oam	1	100%
oobfc	12	100%
packing	2	100%
policier	13	100%
port	105	100%
proxy	7	100%
ptp	77	100%
qos	1	100%
riot	44	100%
rtag7	2	100%
rx	22	100%
sat	29	100%
stat	50	100%
stg	13	100%
swtich	14	100%
time	13	100%
trill	3	100%
trunk	61	100%
tunnel	7	100%
subport	7	100%
udf	6	100%
vlan	106	100%

vxlan	96	100%
Security Totals	1821 tests	100% pass rate

Section 7.6: PHY Test Results

The tables below represent specific results from switch and PHY interoperability and regression testing for the release.

Section 7.6.1: SQA External PHY

Switch Device	*External Phy Device*	*Total Tests*	*% Fail*
56860_A0	phy82780_10G	37	0.00%
56860_A0	phy82780_40G	37	0.00%
56860_A0	phy82792_10GPt_NonEnzo	174	0.00%
56860_A0	phy82792_40GPt_NonEnzo	174	0.00%
56867_A1	phy82764_10G	174	0.00%
56867_A1	phy82764_40G	174	0.00%
56867_A1	phy82764_40HG	174	0.00%
56867_A1	phy82764_42HG	174	0.00%
56867_A1	phy82332_100GPt	26	0.00%
56867_A1	phy82332_100G_gbox	26	0.00%
56867_A1	phy82332_106GPt	26	0.00%
56867_A1	phy82332_40G	26	0.00%
56867_A1	phy82332_10G	26	0.00%
56867_A1	phy82332_11G	26	0.00%
56867_A1	phy82332_1G	26	0.00%
56860_A1	phy82332_retimer_10G	26	0.00%
56860_A1	phy82332_retimer_40G	26	0.00%
56860_A1	phy82332_retimer_100G	26	7.69%
56860_A1	phy82332_100G_gbox_lane0-9	26	0.00%
56960_B0	phy82864_100G	26	0.00%
56960_B0	phy82864_100G_alt	26	0.00%
56960_B0	phy82864_40GPt	26	7.69%
56960_B0	phy82864_40GPt_alt	26	7.69%
56960_B0	phy82864_10G	26	7.69%
56960_B0	phy82864_10G_alt	26	7.69%
56960_B0	phy82864_11G	26	0.00%
56960_B0	phy82864_11G_alt	26	0.00%
56960_B0	phy82864_42GPt	26	0.00%
56960_B0	phy82864_42GPt_alt	26	0.00%
56960_B0	phy82864_106G_alt	26	0.00%

56960_B0	phy82864_40G2x20	26	11.54%
56960_B0	phy82864_40G2x20_alt	26	11.54%
56960_B0	phy82864_42G2x20	26	3.85%
56960_B0	phy82864_40GMux	26	7.69%
56960_B0	phy82864_50G	26	0.00%
56960_B0	phy82864_25G	26	0.00%
56765_B0	phy54140_1G_Copper	174	0.00%
56765_B0	phy54140_1G_Fiber	174	0.00%
56765_B0	phy54190_Copper	7	0.00%

Section 7.6.2: Interop External PHY

P2P Suite & Loopback Suites

Switch Device	*Port Macro*	*Total Tests*	*% Fail*
56560_B0	CONFIG_XE_G40_84328_10G	24	12.50%
56560_B0	CONFIG_XE_G40_84328_1G	7	0.00%
56560_B0	CONFIG_XE_G40_84328_40G	24	12.50%
56560_B0	CONFIG_HG_G40_84328_42G	25	16.00%
56560_B0	CONFIG_XE_MT_84757_10G	19	0.00%
56560_B0	CONFIG_XE_MAKO_1G	12	0.00%
56560_B0	CONFIG_XE_MAKO_RTMR	2	0.00%
56560_B0	CONFIG_XE_KOI_10G	4	0.00%
56560_B0	CONFIG_XE_MAKO_10G	8	0.00%
56560_B0	CONFIG_XE_MAKO_5G	2	0.00%
56560_B0	CONFIG_XE_KOI_1G	12	8.33%
56160_B0	CONFIG_XE_QD28_82780_40G	39	0.00%
56160_B0	CONFIG_XE_QD28_82780_10G	51	23.53%
56160_B0	CONFIG_XE_ORCA_8488X_10G	18	16.67%
56160_B0	CONFIG_XE_ORCA_8488X_2P5G	2	0.00%

56760_A0	CONFIG_XE_MT_84744_10G	32	6.25%
56760_A0	CONFIG_XE_MT_84744_40G	19	0.00%
56760_A0	CONFIG_XE_G28_82322_10G	24	0.00%
56760_A0	CONFIG_XE_ORCA_84888_RETIMER	2	0.00%
56760_A0	CONFIG_XE_G28_82322_40G	24	0.00%
56760_A0	CONFIG_XE_54140_COPPER	18	5.56%
56760_A0	CONFIG_XE_KOI_8485X	12	0.00%
56760_A0	CONFIG_XE_ORCA_84888	18	0.00%
56760_A0	CONFIG_XE_54140_FIBER	8	0.00%
56860_A0	CONFIG_HG_DINO_82332_106G_GB_L0_9	35	0.00%
56860_A0	CONFIG_CE_DINO_82332_100G_PT	80	10.00%
56860_A0	CONFIG_CE_DINO_82332_100G_PT_L2_11	80	10.00%
56860_A0	CONFIG_XE_DINO_82332_40G_10G_MIXEDMODE	80	25.00%
56860_A0	CONFIG_HG_DINO_82332_11G_RETIMER	66	12.12%
56860_A0	CONFIG_XE_DINO_82332_40G_RETIMER	83	26.51%
56860_A0	CONFIG_CE_DINO_82332_100G_GB_L1_10	113	18.58%
56860_A0	CONFIG_CE_DINO_82332_100G_GB_L2_11	113	17.70%
56860_A0	CONFIG_CE_DINO_82332_100G_RETIMER	80	10.00%
56860_A0	CONFIG_XE_QD28_82780_40G	48	0.00%
56860_A0	CONFIG_HG_DINO_82332_42G	53	9.43%
56860_A0	CONFIG_XE_SESTO_82764_10G_PT	54	0.00%
56860_A0	CONFIG_HG_DINO_82332_10G	22	0.00%
56860_A0	CONFIG_GE_DINO_82332_1G_RETIMER	56	57.14%
56860_A0	CONFIG_HG_DINO_82332_106G_GB	88	0.00%
56860_A0	CONFIG_CE_SESTO_82792_100G_343	78	0.00%
56860_A0	CONFIG_XE_QD28_82780_10G	93	2.15%
56860_A0	CONFIG_XE_DINO_82332_40G	83	24.10%
56860_A0	CONFIG_XE_DINO_82332_10G_RETIMER	80	25.00%
56860_A0	CONFIG_XE_SESTO_82764_40G_MUX	45	0.00%

56860_A0	CONFIG_XE_DINO_82332_40G_AN	19	0.00%
56860_A0	CONFIG_XE_G40_84328_10G	36	8.33%
56860_A0	CONFIG_HG_DINO_82332_106G_PT	100	1.00%
56860_A0	CONFIG_XE_SESTO_82764_40G_PT	60	0.00%
56860_A0	CONFIG_CE_DINO_82332_100G_PT_L1_10	80	10.00%
56860_A0	CONFIG_XE_DINO_82332_10G	129	20.93%
56860_A0	CONFIG_XE_G40_84328_40G	36	5.56%
56860_A0	CONFIG_HG_DINO_82332_11G	35	11.43%
56860_A0	CONFIG_CE_DINO_82332_100G_GB	129	9.30%
56860_A0	CONFIG_XE_54210	18	0.00%
56860_A0	CONFIG_XE_G40_84328_1G	10	0.00%

Section 7.6.3: Interop Internal PHY

P2P Suite & Loopback Suites

Switch Device	*Port Macro*	*Total Tests*	*% Fail*
56560_B0	CONFIG_XE_56170_TSCF_25G	5	0.00%
56560_B0	CONFIG_XE_56170_TSCE_10G_VCO6250	9	0.00%
56560_B0	CONFIG_XE_56170_TSCE_2p5G	30	0.00%
56560_B0	CONFIG_XE_56170_TSCE_2p5G_DPLL	18	0.00%
56560_B0	CONFIG_XE_56170_TSCE_11GHG	84	0.00%
56560_B0	CONFIG_XE_56170_TSCE_10G_DPLL	39	0.00%
56560_B0	CONFIG_XE_56170_Viper_2p5G_fiber	24	8.33%
56560_B0	CONFIG_XE_56170_TSCE_5G_DPLL	9	0.00%

56560_B0	CONFIG_XE_56170_QTCE_2p5G	36	0.00%
56560_B0	CONFIG_XE_56170_TSCE_TSCF_40G	69	0.00%
56560_B0	CONFIG_XE_56170_TSCE_10G	66	0.00%
56560_B0	CONFIG_XE_56170_TSCE_TSCF_42GHG	30	0.00%
56560_B0	CONFIG_XE_56170_TSCE_TSCF_10GHG	51	0.00%
56560_B0	CONFIG_XE_56170_TSCE_40G	110	0.00%
56560_B0	CONFIG_XE_56170_TSCF_25GHG	10	0.00%
56560_B0	CONFIG_XE_56170_TSCE_TSCF_10G	72	0.00%
56560_B0	CONFIG_XE_56170_Viper_2p5G	21	0.00%
56560_B0	CONFIG_XE_56170_TSCF_50GHG	35	0.00%
56560_B0	CONFIG_XE_56170_TSCE_42GHG	42	0.00%
56560_B0	CONFIG_XE_56170_TSCF_10G	25	0.00%
56560_B0	CONFIG_XE_56170_TSCF_10GHG	25	0.00%
56160_B0	CONFIG_QSGMII_TSCE	70	2.86%
56160_B0	CONFIG_QSGMII_QTC	44	0.00%
56160_B0	CONFIG_SGMII_QTC	71	0.00%
56160_B0	CONFIG_QSGMII_GPHY	44	0.00%
56160_B0	CONFIG_SGMII_GPHY	44	0.00%
56160_B0	CONFIG_SGMII_TSCE	20	0.00%
56760_A0	CONFIG_HG_72X11	52	0.00%
56760_A0	CONFIG_XE_16X10	54	0.00%
56760_A0	CONFIG_HG_TSCF_X2	252	3.17%
56760_A0	CONFIG_XE_TSCE_X2	68	11.76%
56760_A0	CONFIG_XE_TSCE_X1	135	0.00%
56760_A0	CONFIG_HG_18X42	60	0.00%
56760_A0	CONFIG_HG_TSCE_X1	48	0.00%
56760_A0	CONFIG_XE_18X40	88	3.41%
56760_A0	CONFIG_XE_TSCF_X2	59	20.34%
56760_A0	CONFIG_HG_TSCF_X4	216	3.24%

56760_A0	CONFIG_HG_TSCE_X2	48	0.00%
56760_A0	CONFIG_XE_16X25	59	20.34%
56760_A0	CONFIG_HG_16X27	84	9.52%
56760_A0	CONFIG_XE_TSCF_X4	159	2.52%
56760_A0	CONFIG_XE_72X10	48	0.00%
56760_A0	CONFIG_XE_TSCF_X2_40G	118	13.56%
56270_A0	CONFIG_XE_56270_8x10G_4x2p5G_copper	12	0.00%
56270_A0	CONFIG_XE_56270_8x10G_4x2p5G_fiber	24	0.00%
56270_A0	CONFIG_XE_56270_8x10G_4x2p5G_fiber	24	0.00%
56270_A0	CONFIG_XE_56270_8x10G_4x2p5G_copper	12	0.00%
56965_A1	CONFIG_XE_MSA_50G	12	0.00%
56965_A1	CONFIG_XE_MSA_25G	12	0.00%
56970_B0	CONFIG_XE_TSCE_X1	56	0.00%
56970_B0	CONFIG_XE_64XD50	241	7.05%
56970_B0	CONFIG_XE_128X1	25	0.00%
56970_B0	CONFIG_HG_32X106	152	0.00%
56970_B0	CONFIG_XE_MSA_25G	12	0.00%
56970_B0	CONFIG_XE_128X10	88	2.27%
56970_B0	CONFIG_XE_MSA_50G	6	0.00%
56970_B0	CONFIG_XE_32X100	198	2.02%
56970_B0	CONFIG_XE_128X25	115	3.48%
56970_B0	CONFIG_HG_128X27	105	11.43%
56970_B0	CONFIG_HG_64XD53	210	4.29%
56860_A0	CONFIG_CE_8x100_442	84	13.10%
56860_A0	CONFIG_HG_8x100_343	24	4.17%
56860_A0	CONFIG_XE_32X40	128	4.69%
56860_A0	CONFIG_CE_8X100_343_IEEE	84	13.10%
56860_A0	CONFIG_HG_104x10	24	0.00%
56860_A0	CONFIG_HG_32X42	60	0.00%

56860_A0	CONFIG_XE_104x10	111	0.00%

Section 7.6.3: 88060 PHY

Phy 88060 Results

Switch Device	*Phy Config*	*Total Tests*	*% Fail*
56960_A0	phy88060	133	2.14%
56768_B0	phy88060	73	2.74%
56850_A2	phy88060	73	2.74%

Section 7.7: Static Code Analysis

Below shows our static analysis backlog for devices supported in the 6.5.x releases:

Section 7.7.1: Unresolved Static Code Analysis Issues

Area	*Open Issues SDK 6.5.12*	*Open Issues SDK 6.5.11*	*Open Issues SDK 6.5.10*	*Open Issues SDK 6.5.9*	*Open Issues SDK 6.5.8*	*Open Issues SDK 6.5.7*	*Open Issues SDK 6.5.6*	*Open Issues SDK 6.5.5*
DNX	1	0	7	8	11	47	63	124
XGS	1	2	12	6	25	10	15	23
SerDes	5	6	6	10	12	10	38	25
Common	3	3	4	8	3	4	27	32
Total	10	11	29	32	51	71	143	204

Section 8: Service Impacting Defects

A Service Impacting Defect (SID) is any defect (internal or external) that has high potential to severely disrupt network operations in a deployed system. The following table lists SIDs identified since our last SDK release.

Table 5: Resolved Service Impacting Defects

Reference	*Chips*	*Affected Versions*	*Errata Synopsis*	*Details*
SDK-131574	56850_A2, 56860_A0	6.5.5, 6.5.6, 6.5.7, 6.5.8, 6.5.9, 6.5.10, 6.5.11	SER error on L3_DEFIP_ALPM_IPV4 table during SBUS insert operation	The SDK inline recovery of ALPM tables doesn't cover XOR bank error case. This could cause insert/lookup/delete op for ALPM tables fails when SER error exists on XOR bank.

Section 9: Potential Security Vulnerabilities

Broadcom treats security vulnerability issues reported by customer Product Security Incident Response Teams (PSIRT) with very high importance and urgency. Please ensure that any such issues reported and filed by your organization through the Broadcom customer support portal specifically use the acronym “PSIRT” in the CSP case summary and/or description. This will allow the Broadcom engineering teams to track, analyze, and address these issues as quickly as possible.

Starting in SDK 6.5.10 we have updated our release process to communicate potential security vulnerabilities in our software products. The list below are issues we have found and handled in this release for customer awareness.

Table 6: Security Vulnerabilities

Reference	*Chips*	*Affected Versions*	*Errata Synopsis*	*Details*
None in this release				

Section 10: GNU tools versions

Broadcom uses GNU tools, specifically “gmake”, “gcc”, several Linux distributions and Linux kernel versions for SDK build and validation in-house. The following table summarizes the tools used in this release

Table 7: GNU tools versions

CPU	*gmake*	*gcc*	*Operating System*	*Linux Kernel*
RSX	4.1	4.7.2	Broadcom LDK 3.5.5	3.6.5
SLK	4.1	4.9.2	Broadcom LDK 4.1.10	3.14.65
iProc	4.1	4.7.2	Broadcom XLDK 4.0.1	4.4.39
XLR	4.1	4.9.2	Fedora Linux 2.1	4.4.6
GTS	4.1	4.9.2	Fedora Linux 2.1	4.4.6

If there are any issues with running or compiling SDK with GCC versions higher than what is listed above, such issues should be reported via Broadcom Customer Support for evaluation. If the issue is caused by SDK coding or logic error, it will be resolved in a subsequent SDK release.

However, if the issue is caused by the nature of how new versions of GCC handle compilation and is not directly related to SDK coding or logic errors, it will be fixed on best-effort basis, with no guarantee for a specific ETA.

Section 11: Resolved and Unresolved Issues for 6.5.12

Section 11.1: Resolved Issues and Improvements

Starting with SDK 6.5.9, Broadcom provides a list of resolved improvements and defects in a sortable spreadsheet format rather than a static table in this document. For the full resolved list, please reference the file `SDK-6.5.x-Resolved-Issues-Improvements.xlsx` in the RELDOCS directory in the release package.

Section 11.2: Unresolved Issues

The following open Urgent priority issues remain unresolved in SDK 6.5.12. These are in process of being evaluated for inclusion in a future SDK release:

Number	*CSP*	*Chips*	*Errata For 6.5.12*
PHY-2902	1123900	84756_A0 84756_C0	SDK-6.5.7 fails to download firmware to PHY 84756
PHY-2986		56760_A0	The port speed was changed to 40G if change the encap from Higig to Eth
PHY-3003	1139654	56846_A0 56846_A1	LINK state issue with BCM84834
PHY-3217	CS3833597	56340_A0 56342_A0 56344_A0	EEE RxDuration counter do not increment after port disable, speed set and enable it
SDK-126459	CS3127642	56850_A0	CLONE - [56850]internal loopback port bitmap usage in FP
SDK-135037	CS3744866	AllChips	Inports mask in ACL entry is not consistent after doing fanout
SDK-135807	CS3742326	88470_A0	PRGE program ARAD_EGR_PROG_EDITOR_PROG_BFD_ACC_WA should remove the UDH
SDK-136189	CS3938770	53461_A0 56270_A0	Metrolite: Maximum number of ports per trunk group isn't same as mentioned in Data Sheet of BCM56270
SDK-118159		88670_A0 88680_A0	Add SW WA for PVT-mon issue - phymod API

Section 12: Compatibility

Section 12.1: Broadcom Embedded Applications Firmware Compatibility Matrix

The following table shows new feature support added in Firmware releases for switch devices compatible with the corresponding SDK release. Please refer to the appropriate Network Switching SDK Firmware release notes publication (56XX0_88XX0_FW-RNxxx-R) for the indicated version below for full details. Note there was no new device support added in SDK 6.5.8 and SDK 6.5.11.

	SDK-6.5.12	SDK-6.5.10	SDK-6.5.9	SDK-6.5.7	SDK-6.5.6
4.3.6	BCM56870 BCM53570				
4.3.5		BCM88270 BCM56960 BCM56970			
4.3.4			BCM88270		
4.3.3				BCM56760 BCM56565	BCM56270 BCM56760 BCM56565

Section 12.2: BMACSEC/iMACSEC SDK Compatibility Matrix

Note iMACSEC is specifically for use with the BCM54190 integrated PHY driver.

Switch SDK Release	*BMACSEC Release*	*iMACSEC Release*
6.5.7	4.16	1.0
6.5.8	4.16	1.1
6.5.9	4.16	1.1
6.5.10	4.16	1.2
6.5.11	4.17	1.2
6.5.12	4.17	1.2

Section 12.3: PHY Firmware Compatibility Matrix

The following table identifies changes in PHY firmware for newer PHY devices and for the serdes core.

PHY Core	**6.5.7 Firmware Versions**	**6.5.8 Firmware Versions**	**6.5.9 Firmware Versions**	**6.5.10 Firmware Versions**	**6.5.11 Firmware Versions**	**6.5.12 Firmware Versions**
BCM84888	00.00.10	00.00.10	A0: 1.00.03 B0: 2.00.03	A0: 1.00.03 B0: 2.00.03	A0: 1.00.09 B0: 2.00.09	A0: 1.00.09 B0: 2.00.09
BCM84858	01.03.02	01.03.04	01.03.04	01.03.04	01.03.04	01.03.04
BCM84856	01.03.02	01.03.04	01.03.04	01.03.04	01.03.04	01.03.04
Eagle dual PLL						D10F_13
Falcon	D10B_13	D10B_13	D10B_14	D10B_14	D10B_14	D10B_14
Falcon dual PLL						D10B_19

Falcon16						D103_04
Merlin			D101_0C	D101_0C	D101_0C	D101_0C

Section 12.5: SDK and BCM88060 FW Compatibility Matrix

The firmware binary is part of the SDK release. Below table shows the firmware version compatible with which SDK release.

Switch SDK Release	***88060 FW version***
6.5.10	1.0.10
6.5.11	1.0.11
6.5.12	1.0.12

Section 13: SDK Externally Licensed Software Components

The SDK contains a number of third-party externally licensed software components. This appendix contains information regarding these components, the license for each of these components, and where these components are used in SDK.

Component	***Origin***	***Location in Source Tree***
EDITLINE	/afs/athena.mit.edu/contrib/sipb/src/editline	src/sal/appl/editline
LIBXML2	http://xmlsoft.org/downloads.html	src/shared/libxml
ED Editor	USENET comp.sources.misc Volume 9, Issue 36	src/appl/diag/edline.c
CINT	http://www.gnu.org/software/bison/	src/appl/cint/cint_parser.[ch]
BIGDIGITS	David Ireland, copyright (c) 2001-11 by D.I. Management Services Pty Limited <www.di-mgt.com.au>	src/soc/dpp/SAND/Utils/sand_u64.c
APIMODE	http://www.gnu.org/software/bison/	src/appl/diag/api/api_grammar.tab.[ch]

VxWorks	Wind River Systems, Inc.	systems/vxworks
SFlow	http://www.inmon.com/technology/sflowlicense.txt	N/A - see Section 13.8

Section 13.1: EDITLINE License terms and conditions

```
--------------------------------------------------------------------------------

This package was obtained in 1999 and modified to fit the Broadcom
SDK. In 2015 is was modified further to perform terminal I/O through
call-backs, and several unused FSF compatibility functions were
removed. For SDK purposes, the library can still be replaced by the
FSF readline library.

The original library is maintained at GitHub:
https://github.com/troglobit/editline

--------------------------------------------------------------------------------

ORIGINAL DESCRIPTION

This is a line-editing library.  It can be linked into almost any
program to provide command-line editing and recall.

It is call-compatible with the FSF readline library, but it is a
fraction of the size (and offers fewer features).  It does not use
standard I/O.  It is distributed under a "C News-like" copyright.
```

```
-------------------------------------------------------------------------------

ORIGINAL COPYRIGHT

 Copyright 1992,1993 Simmule Turner and Rich Salz.  All rights reserved.

 This software is not subject to any license of the American Telephone
 and Telegraph Company or of the Regents of the University of California.

 Permission is granted to anyone to use this software for any purpose on
 any computer system, and to alter it and redistribute it freely, subject
 to the following restrictions:
 1. The authors are not responsible for the consequences of use of this
    software, no matter how awful, even if they arise from flaws in it.
 2. The origin of this software must not be misrepresented, either by
    explicit claim or by omission.  Since few users ever read sources,
    credits must appear in the documentation.
 3. Altered versions must be plainly marked as such, and must not be
    misrepresented as being the original software.  Since few users
    ever read sources, credits must appear in the documentation.
 4. This notice may not be removed or altered.

-------------------------------------------------------------------------------
```

Section 13.2: LIBXML2 - XML C parser terms and conditions

Package was obtained from http://xmlsoft.org/ and is used by diagnostics tool for miscellaneous input/output tasks

This README is part of SDK under src/shared/libxml and is as follows:

```
/*
```

```
 * $Id$
 *
 * $Copyright: (c) 2011 Broadcom Corporation
 * All Rights Reserved.$
 */
```

This package was obtained from http://xmlsoft.org/downloads.html
(ftp://xmlsoft.org/libxml2/libxml2-2.7.2.tar.gz)
and was modified for purposes of inclusion into the SOC diagnostics shell.

Only certain portion of package was included in SDK in 2 places:

Under srs/shared/libxml

chvalid.c, config.h, dict.c, encoding.c, entities.c, error.c
globals.c, hash.c, libxml.h, list.c, Makefile, parser.c
parserInternals.c, SAX2.c, threads.c, tree.c, uri.c, valid.c
xmlIO.c, xmlmemory.c, xmlsave.c, xmlstring.c, xmlunicode.c

Under include/shared/libxml

catalog.h, chvalid.h, debugXML.h, dict.h, DOCBparser.h
encoding.h, entities.h, globals.h, hash.h, HTMLparser.h
HTMLtree.h, list.h, parser.h, parserInternals.h, pattern.h
relaxng, SAX2.h, threads.h, tree.h, uri.h, valid.h, xinclude.h
xlink.h, xmlautomata.h, xmlerror.h, xmlexports.h, xmlIO.h
xmlmemory.h, xmlmodule.h, xmlregexp.h, xmlsave.h, xmlstring.h
xmlunicode.h, xmlversion.h, xpath.h, xpathInternals.h, xpointer.h

No functionality was changed, but there were modifications to match SDK requirements

Copyright

Section 13.3: ED Editor License terms and conditions

```
ed - standard editor
  ^^
    Authors: Brian Beattie, Kees Bot, and others

 Copyright 1987 Brian Beattie Rights Reserved.
 Permission to copy or distribute granted under the following conditions:
 1). No charge may be made other than reasonable charges for reproduction.
 2). This notice must remain intact.
 3). No further restrictions may be added.

 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  TurboC mods and cleanup 8/17/88 RAMontante.
  Further information (posting headers, etc.) at end of file.
 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
 Modification log:
 25Aug92  (W.Metzenthen) Changed malloc() call to calloc() in makebitmap()
          to remove bugs under Linux. Changed a few '^' to the correct '~'.
          General tidying. Recognize Linux via the __linux__ symbol.
          Main change based upon suggestion by Wolfgang Thiel.
 07Sep99  Changed large amounts of stuff to simplify  --Curt McDowell
```

Section 13.4: CINT parser license terms and conditions

The C code for the CINT parser was generated by using GNU Bison parser generator from the file cint_grammar.y CINT is an optional diagnostic tool that can be included in your system by adding CINT to the FEATURE_LIST in SDK compilation flags.

Removed files:

None

Added files:

None

Changed functionality:

None

```
--------------------------------------------------------------------------------
/* A Bison parser, made by GNU Bison 2.4.1.  */

/* Skeleton implementation for Bison's Yacc-like parsers in C

      Copyright (C) 1984, 1989, 1990, 2000, 2001, 2002, 2003, 2004, 2005, 2006
   Free Software Foundation, Inc.

   This program is free software: you can redistribute it and/or modify
```

```
   it under the terms of the GNU General Public License as published by
   the Free Software Foundation, either version 3 of the License, or
   (at your option) any later version.

   This program is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
   GNU General Public License for more details.

   You should have received a copy of the GNU General Public License
   along with this program.  If not, see <http://www.gnu.org/licenses/>.  */

/* As a special exception, you may create a larger work that contains
   part or all of the Bison parser skeleton and distribute that work
   under terms of your choice, so long as that work isn't itself a
   parser generator using the skeleton or a modified version thereof
   as a parser skeleton.  Alternatively, if you modify or redistribute
   the parser skeleton itself, you may (at your option) remove this
   special exception, which will cause the skeleton and the resulting
   Bison output files to be licensed under the GNU General Public
   License without this special exception.

   This special exception was added by the Free Software Foundation in
   version 2.2 of Bison.  */

/* C LALR(1) parser skeleton written by Richard Stallman, by
   simplifying the original so-called "semantic" parser.  */
```

Section 13.5: BIGDIGITS license terms and conditions

```
Contains BIGDIGITS multiple-precision arithmetic code originally
written by David Ireland, copyright (c) 2001-11 by D.I. Management
Services Pty Limited <www.di-mgt.com.au>, and is used with
permission.

David Ireland and DI Management Services Pty Limited make no
representations concerning either the merchantability of this software
or the suitability of this software for any particular purpose. It is
provided "as is" without express or implied warranty of any kind. Our
liability will be limited exclusively to the refund of the money you
paid us for the software, namely nothing. By using the software you
expressly agree to such a waiver. If you do not agree to the terms, do
not use the software.
```

Section 13.6: APIMODE parser license terms and conditions

The C code for the APIMODE parser was generated by using GNU Bison parser generator from the file api_grammar.y APIMODE is an optional diagnostics

shell interface that can be included in your system by adding APIMODE to the FEATURE_LIST in SDK compilation flags.

See “CINT parser license terms and conditions” for the Bison licence.

Section 13.7: Wind River Systems license terms and conditions

See WRS_LICENSE.pdf contained in each systems/vxworks subdirectory.

Section 13.8: SFlow license terms and conditions

Broadcom provides several API modules that refer to SFlow by name, specifically Field, Mirror, Port, and Switch. All are implemented as per IETF RFC-3176. Please review the separate sflowlicense.txt file for terms of the agreement used by Broadcom in our implementation.