DNX28/DNX16

Traffic Manager Software Compatibility Guide

Application Note
BCM88670
BCM88690
BCM88800
BCM88480
BCM88280
BCM88830

Table of Contents

Chapter 1: Introduction

1.1 Purpose and Audience

This document summarizes the differences in the BCM SDK Traffic Manager Software interface between the BCM88670-family devices and the BCM88690-family devices. For detailed documentation about the BCM88690, refer to *Traffic Management Architecture* (88690-DG1xx) and the *Traffic Manager Programming Guide* (88690-PG2xx).

For differences in the BCM SDK Packet Processor Software interface between the BCM88670 and the BCM88690 devices, refer to the *Packet Processor Software Compatibility Guide* (88670-88690-AN2xx).

This document is intended for customers using the BCM SDK to configure the BCM88670 family of devices. The goal of this document is to assist the migration process from existing applications that configure the BCM88670 family of devices to applications that configure the BCM88690 family of devices.

NOTE: The BCM SDK Traffic Manager Software interface for BCM88680 and BCM88670 devices is identical, with a few exceptions. All references to the BCM88670 in the scope of this document are also relevant for BCM88680 devices and derivative SKUs, unless explicitly stated otherwise.

Additionally, the BCM SDK Traffic Manager Software interface for the BCM88690, BCM88800, BCM88480, BCM88280, and BCM88830 devices is identical, with a few exceptions. All references to the BCM88690 in the scope of this document are also relevant for BCM88800 and BCM88480 devices and derivative SKUs, unless explicitly stated otherwise.

This document does not provide the theory of operation and driver reference of the supported features. They are documented in the BCM88690 *Traffic Manager Programming Guide* (88690-PG2xx).

1.2 SW Interface Changes

1.2.1 Scope and Motivation

Preserving the software interface of the BCM88670 device driver was a key consideration in the BCM88690 device driver design. The software interface was modified in the following cases:

- Significantly modified or extended device functionality
- Significantly modified hardware implementation, affecting the required configuration sequence
- Improved software design resulting in a significantly more intuitive and efficient API

In some cases, the existing interface support was preserved in parallel to introducing a new interface. In these cases, the existing interface allows configuring a subset of features shared by both the BCM88670 and BCM88690 devices or provides a backward-compatible API in parallel to a more intuitive and efficient API.

1.2.2 Interface Modification Types

Changes in the software interface fall into one of the following categories:

- API prototype changes. Changes in the BCM API prototype definition, for example:
  - Added, removed, or modified API functions.
  - Added, removed, or modified parameters of an existing API.
- Calling sequence changes. Changes in the required configuration sequence, for example:
  - Using existing APIs in a different way.
- SOC property changes. Changes in the static SOC configuration properties, for example:
  - New, removed, or modified SOC properties.
  - A different interpretation of existing SOC properties.

For API prototype changes, this document indicates the BCM88690 API and the related BCM88670 API and explains the differences between them. If the legacy API is supported for backward compatibility, this document describes the limitations that apply when using the legacy API.

For calling sequence changes, this document points out the required changes in the API calling sequence. A configuration sequence can also be referenced in the relevant cint example sequence, as mentioned in the corresponding section of the *Traffic Manager Programming Guide* (88690-PG2xx).

For SOC property changes, this document indicates the BCM88690 SOC property and the related BCM88670 API SOC property. If the legacy property is also supported for backward compatibility, this document describes the limitations that apply when using the legacy property.

Chapter 2: Compatibility Highlights

This section highlights major changes in the SDK. For full details, see the sections in this document related to the change. Important changes include the following:

- Initialization sequence:
  - `soc_init` should be removed from the initialization sequence.
- Port configuration:
  - Explicitly setting the speed (using the SOC property or API for the dynamic port) is now mandatory.
- Network interface (NIF) and PHYs:
  - NIF PHYs are 0-based instead of 1-based (affects the polarity SOC property, `bcm_port_add` API, and several diagnostics commands).
  - The following items were changed to provide better support for Blackhawk (BH) PHYs. Moreover, to have the same APIs for all of the PHYs, the new API format was also applied to Falcon PHYs on the devices of the new generation (BCM8848X and BCM8880X):
    - Lane swap and polarity: APIs and SOC property changed.
    - APIs to configure speed, interface, FEC, link training, and FW modes are deprecated. Instead, use `bcm_port_resource_set\bcm_port_resource_multi_set`.
    - A new API for TX taps.
    - A new set of APIs for autonegotiation abilities advertising.
- Ingress queuing:
  - Direct mapping mode is not supported.
  - When creating a queue system port, gport is mandatory (modport cannot be used for these APIs).
  - Credit request profiles (also referred to as *delay tolerance*):
    - No default profiles are configured during initialization.
    - A cint example for configuring legacy profiles is provided.
  - Implicit setting of VOQ rate class thresholds (using `UCAST_QUEUE_GROUP` gport) is not supported. All VOQ rate class thresholds can be set using a rate class gport only.
  - Setting queues ranges (unicast or multicast) is now performed by a single control: `bcmFabricMulticastQueueMax`.
- Multicast:
  - Some redundant APIs are deprecated. See Chapter 10, Multicast.
- Sniff (mirror):
  - The sequence has changed: The port for mirroring should be explicitly created, and each forwarding port should be assigned a mirror port.
- Egress traffic management:
  - RCY interface: The BCM88690 has two RCY interfaces. When creating a recycle port, indicate its interface.
  - The header compensation mechanism has changed. For details, see Chapter 14, Egress Traffic Management and refer to the *Traffic Manager Programming Guide* (88690-PG2xx).
- Counter processors:
  - The sequence has completely changed. Refer to the *Traffic Manager Programming Guide* for details.
- Metering:
  - Major changes are summarized in Chapter 19, Metering. For additional information, read the Metering section in the *Traffic Manager Programming Guide*.

Chapter 3: Initialization Template

3.1 Calling Sequence

3.1.1 Initialization Sequence

The `soc_init()` API is not supported for the BCM88690. The step calling to `soc_init()` should be removed from the initialization sequence.

3.1.2 Deinitialization Sequence

The `soc_deinit()` API is not supported by the BCM88690. The step calling to `soc_deinit()` should be removed from the deinitialization sequence.

3.1.3 Reference Application Example

The reference application example and its shell command, `init_dnx` were redesigned and developed from scratch.

Refer to the *Traffic Manager Programming Guide* Initialization Sequence section, for more information and usage.

3.2 SOC Properties

If the same configuration file is used by several devices, it is possible to distinguish the configuration by using a special suffix. The supported suffix options for the DNX16 devices have changed as compared to the DNX28 devices. The supported options are listed in the *Traffic Manager Programming Guide (88690-PG2xx),* Initialization Sequence section.

3.2.1 device_core_mode

BCM88690 does not support the SOC property.

Chapter 4: Port Provisioning

4.1 APIs

`bcm_port_add()` and `bcm_port_get()`

- For the BCM88690 device only: TDM and ILKN data ports are not supported.
- `interface_info.phy_port` is not supported.
  Instead, for Ethernet ports, `interface_info.phy_pbmp` should be set.
  Note that PHYs in the bitmap are 0-based (for example, for the BCM8890, the valid bits are 0 to 95).
- `mapping_info.core` is not flexible for NIF interface ports and must be set according to the NIF interface table in *Traffic Manager Programming Guide*.
- `interface_info.interface` should be equal to `BCM_PORT_IF_ETH`.
  – Legacy interfaces are still supported but will have the same impact as using `BCM_PORT_IF_ETH`.
- No default speed exists. Therefore, after adding a port, calling `bcm_port_resource_set` or `bcm_port_resource_multi_set` is mandatory.

4.2 SOC Properties

`custom_feature_dynamic_port`

- The SOC property is not supported by the BCM88690.
- Dynamic ports will be allowed in any mode.

Chapter 5: Network Interface and PHY

5.1 NIF PHY Indexing

The NIF PHY index is 0-based and not 1-based, as in previous devices.

This affects the polarity SOC property, `bcm_port_add` API, and several diagnostic commands.

5.2 Port to Core Mapping

5.2.1 SOC Properties

The assignment of lanes to the core in the BCM88690 is hard-wired; therefore, the `core_id` information in the `ucode_port` SOC property is redundant.

The SOC property format remains the same as in the BCM88670. However, trying to assign a link to the wrong core will return an error.

5.3 Lane Swap and Polarity

5.3.1 APIs

Lane swap configuration in the BCM88670 was supported by the `BCM_PORT_PHY_CONTROL_LANE_SWAP` control.

This control is no longer supported in the BCM88690 generation of devices, so to configure lane swapping, use the `bcm_port_lane_to_serdes_map_set` and `bcm_port_lane_to_serdes_map_get` APIs. For more information, refer to the Lane Map section in the *Traffic Manager Programming Guide*.

5.3.2 SOC Properties

Lane swap configuration per quad is not supported. Instead, use lane swap configuration per lane. This means the old format of the BCM88670:

```
phy_tx_lane_map.quad<quad_id>=0x<lane3_map><lane2_map><lane1_map><lane0_map>
```

was replaced with the following new format:

```
lane_to_serdes_map_fabric_lane<lane_id>=rx<srd_rx_id>:tx<srd_tx_id>
```

For more information, refer to the Lane Map section in the *Traffic Manager Programming Guide*.

Additionally, `phy_tx_polarity_flip_phy` and `phy_rx_polarity_flip_phy` are 0-based in BCM88690 (these properties were 1-based in the BCM88670 family).

5.4 PM Configuration

5.4.1 APIs

Several PM configurations that were supported with several APIs and SOC properties in the BCM88670 are now supported with the APIs `bcm_port_resource_set` and `bcm_port_resource_get` (or `bcm_port_resource_multi_set` and `bcm_port_resource_multi_get` for a group of ports). This is true for both Blackhawk and Falcon PHYs. The following sections describe the deprecated configurations and their new API.

5.4.1.1 Speed

The `bcm_port_speed_set` and `bcm_port_speed_get` APIs are no longer supported in the BCM88690 and later devices.

Use the `bcm_port_resource_set` and `bcm_port_resource_get` APIs to set and get the speed (or `bcm_port_resource_multi_set` and `bcm_port_resource_multi_get` for a group of ports).

5.4.1.2 Interface

The `bcm_port_interface_set` API and the PHY control `BCM_PORT_PHY_CONTROL_INTERFACE` are not supported in the BCM88690 and later devices.

Use the `bcm_port_resource_set` API to align various interface properties (or `bcm_port_resource_multi_set` for a group of ports).

5.4.1.3 Link Training

The PHY control `BCM_PORT_PHY_CONTROL_CL72` is not supported in the BCM88690 and later devices.

Use the `bcm_port_resource_set` API to enable and disable link training (or `bcm_port_resource_multi_set` for a group of ports).

5.4.1.4 Forward Error Correction

In the BCM88690 and later devices, FEC selection is no longer controlled using `bcm_port_phy_control_set` with the controls:

- `BCM_PORT_PHY_CONTROL_FORWARD_ERROR_CORRECTION`
- `BCM_PORT_PHY_CONTROL_FORWARD_ERROR_CORRECTION_CL91`

Instead, use the `bcm_port_resource_set` API to enable and disable and select the FEC (or `bcm_port_resource_multi_set` for a group of ports).

5.4.1.5 Firmware Controls

The following PHY controls are no longer supported in the BCM88690 and later devices:

- `BCM_PORT_PHY_CONTROL_FIRMWARE_BR_DFE_ENABLE`
- `BCM_PORT_PHY_CONTROL_FIRMWARE_DFE_ENABLE`
- `BCM_PORT_PHY_CONTROL_FIRMWARE_LP_DFE_ENABLE`
- `BCM_PORT_PHY_CONTROL_FIRMWARE_CL72_RESTART_TIMEOUT_ENABLE`
- `BCM_PORT_PHY_CONTROL_FIRMWARE_CL72_AUTO_POLARITY_ENABLE`
- `BCM_PORT_PHY_CONTROL_UNRELIABLE_LOS`
- `BCM_PORT_PHY_CONTROL_MEDIUM_TYPE`
- `BCM_PORT_PHY_CONTROL_FIRMWARE_MODE`

Use the `bcm_port_resource_set` API to set the firmware parameters.

For more information, refer to Ethernet Ports in the Network Interface section of the *Traffic Manager Programming Guide.*

5.4.2 SOC Properties

NOTE: No default speed is provided by the SDK. Therefore, for each port that is defined by the `ucode_port` SOC property, a valid speed must be provided using the `port_init_speed` SOC property

Additionally, the SOC property value –1 for `port_init_speed` is not supported. In previous devices, it allowed the completion of the port initialization without providing the speed.

5.4.2.1 Media Type

The `serdes_fiber_pref` SOC property is not supported in the BCM88690 and later devices.

The media type can be set using the `serdes_lane_config` SOC property.

For more information, refer to the PHY section of the *Traffic Manager Programming Guide* (88690-PG2xx).

5.4.3 Calling Sequence

Starting with the BCM88690, no default speed is provided by the SDK. Therefore, for each port that is added with the `bcm_port_add` API, a valid speed must be provided by using the `bcm_port_resource_set` API before enabling the port. Therefore, adding the NIF port requires the following sequence:

1. `bcm_port_add`
2. `bcm_port_resource_set`
3. `bcm_port_enable_set`

5.5 Transmitter (TX) Controls

5.5.1 APIs

The `BCM_PORT_PHY_CONTROL_DRIVER_CURRENT` control is not supported in the BCM88690.

The following port controls to configure TX FIR parameters are no longer supported in the BCM88690:

- `BCM_PORT_PHY_CONTROL_TX_FIR_PRE`
- `BCM_PORT_PHY_CONTROL_TX_FIR_MAIN`
- `BCM_PORT_PHY_CONTROL_TX_FIR_POST`
- `BCM_PORT_PHY_CONTROL_TX_FIR_POST2`
- `BCM_PORT_PHY_CONTROL_TX_FIR_POST3`

To configure TX FIR parameters, use the `bcm_port_phy_tx_set` API. For more information, refer to the PHY section of the *Traffic Manager Programming Guide*.

5.5.2 SOC Properties

The `serdes_preemphasis` SOC property is not supported in the BCM88690 and later devices.

TX FIR parameters can be set using the `serdes_tx_taps` SOC property. For using this SOC property, refer to the PHY section of the *Traffic Manager Programming Guide*.

The `serdes_driver_current` SOC property is not supported in the BCM88690 and later devices.

5.6 PHY Reset and Squelch Controls

5.6.1 APIs

Reset RX/TX and squelch TX are no longer supported in the BCM88690; therefore, the following PHY controls are no longer supported:

- `BCM_PORT_PHY_CONTROL_RX_RESET`
- `BCM_PORT_PHY_CONTROL_TX_RESET`
- `BCM_PORT_PHY_CONTROL_TX_LANE_SQUELCH`

Use `bcm_port_enable_set` to enable and disable both RX and TX.

NOTE: `BCM_PORT_PHY_CONTROL_RX_LANE_SQUELCH` is supported.

5.7 Load Firmware

5.7.1 SOC Properties

The BCM88690 supports the following changes in regard to the `load_firmware` SOC property:

- Only two suffixes are supported: NIF and Fabric (the BCM88670 supported a suffix per quad).
- The default value for the NIF SOC property was changed to 0x102 (instead of 0x2 in the BCM88670) to add a checksum to the initialization sequence.

5.8 LCPLL Reference Clock

5.8.1 SOC Properties

NOTE: The LCPLL exists on the BCM8869X device only. Later devices, such as the BCM8848X and BCM8880X, do not support an LCPLL, and the reference clock for these devices is fixed to 156.25 MHz. Therefore, the `serdes_<nif/fabric>_clk_freq_in` and `serdes_<nif/fabric>_clk_freq_out` SOC properties are not relevant for the BCM8848X and BCM8880X devices.

The following information is valid for the BCM88690:

- Output LCPLL clocks support 156.25 MHz (312 MHz is also supported, but not for production).
- LCPLL can be bypassed (that is, pass the input reference clock without manipulation). To bypass the LCPLL, set the output reference clock value to *bypass*.
- If the LCPLL is bypassed, the input LCPLL must be set to 156.25 MHz.
- The BCM88690 supports only two NIF LCPLLs, PLL_IDs 0 and 1.

For more information, refer to the PHY section of the *Traffic Manager Programming Guide* (88690-PG2xx).

5.9 Energy Efficient Ethernet

5.9.1 APIs

Energy Efficient Ethernet (EEE) is not supported in the Blackhawk PHYs of all devices (BCM88690 and later). Therefore, the following controls of `bcm_port_control_set` are not supported:

- `bcmPortControlEEEEnable`
- `bcmPortControlEEETransmitIdleTime`
- `bcmPortControlEEETransmitWakeTime`

The BCM88800 and BCM88480 Falcon PHYs support EEE. For details, refer to the *Traffic Manager Programming Guide*.

5.10 NIF Priority

5.10.1 APIs

The new BCM88690 NIF priority API `bcm_port_priority_config` is a much better fit for the BCM88690 silicon. Therefore, it replaces the old API, `bcm_port_nif_priority_set`, which is no longer supported.

Refer to the NIF Priority section of the *Traffic Manager Programming Guide* for details.

5.11 NIF Counters

5.11.1 APIs

The following APIs related to NIF counters are no longer supported in the BCM88690:

- `bcm_port_selective_set` and `bcm_port_selective_get`
- `bcm_port_info_set`, `bcm_port_info_get`, `bcm_port_info_restore`, and `bcm_port_info_save`
- `bcm_stat_init`
- `bcm_stat_get32` and `bcm_stat_multi_get32`
- `bcm_port_stat_get32` and `bcm_port_stat_multi_get32`

5.11.2 SOC Properties

The following SOC properties related to NIF counters are no longer supported in the BCM88690 and later devices:

- `dport_map_enable`
- `bcm_stat_flags`
- `soc_counter_control_level`

The following SOC property was renamed:

`use_fabric_links_for_ilkn_nif_ilknX` was renamed to `ilkn_use_fabric_links_X`.

5.12 NIF Additional APIs

5.12.1 APIs

The following NIF-related APIs are no longer supported in the BCM88690 and later devices:

- `bcm_port_selective_set` and `bcm_port_selective_get`
- `bcm_port_info_set`, `bcm_port_info_get`, `bcm_port_info_restore`, and `bcm_port_info_save`

5.13 Autonegotiation

5.13.1 APIs

The following autonegotiation-related APIs are no longer supported in the BCM88690 and later devices for both Blackhawk and Falcon PHYs:

- `bcm_dnx_port_ability_advert_set` and `bcm_dnx_port_ability_advert_get`
- `bcm_dnx_port_ability_local_get`
- `bcm_dnx_port_ability_remote_get`
- `bcm_port_ability_get`

For the updated sequence, refer to the *Traffic Manager Programming Guide*.

5.14 Port Handling APIs

5.14.1 APIs

The following port handling APIs were removed from the BCM88690 and later devices:

- `bcm_port_init`.
- `bcm_port_clear`.
- `bcm_port_probe`. Use `bcm_port_add` instead. (The API is still supported for fabric ports.)
- `bcm_port_detach` Use `bcm_port_remove` instead. (The API is still are still supported for fabric ports.)

5.15 Port Control API

5.15.1 APIs

Some unused or duplicate controls of the `bcm_port_control` have been removed.

- `bcmPortControlPCS`
- `bcmPortControlLinkDownPowerOn`

5.16 ILKN

5.16.1 General

In the BCM88690, ILKN is to be used only for an ELK connection. There is no support in ILKN ports for data. The BCM88800 and BCM88480 devices support both ILKN ELK and ILKN data.

5.16.2 SOC Properties

The ILKN SOC property changes are as follows:

- In previous devices, the ILKN over fabric SOC property `use_fabric_links_for_ilkn_nif_bmp` was used without any suffix. Starting with the BCM88690, this SOC property is per octet, therefore, it should hold a suffix with the octet ID in the form of `fab_oct<oct_id>` (for example, `use_fabric_links_for_ilkn_nif_bmp_fab_oct0=0xff)`.
- In previous devices, the ILKN over fabric SOC property was named `use_fabric_links_for_ilkn_nif_bmp_fab_octX`. Starting with SDK 6.5.19, the name was changed to `ilkn_use_fabric_links_ilkn_bmp_fab_octX`, for example: `ilkn_use_fabric_links_ilkn_bmp_fab_oct0=0xff`.

  For details, refer to the Interlaken over Fabric section of the *Traffic Manager Programming Guide*.
- The ILKN lane mapping SOC property `Ilkn_lanes` was used with a 24-bit bitmap indicating the active core lanes in the ILKN port.

  Starting with the BCM88690, this SOC property must get, as an input, the device lanes (NIF or fabric) used by the ILKN port, as defined in the `lane_to_serdes_map` SOC property. The value format is a list of lanes separated by commas (or a range separated by a hyphen). The order of lanes has significance; it indicates the ILKN lane order.

NOTE: Both input types are valid for the BCM88690, but with the BCM88800 and BCM88480 devices, only the numeric input type is valid.

Starting with the BCM88690 device, the following ILKN SOC properties were removed:

- `ilkn_counters_mode`
- `ilkn_num_lanes` (use `ilkn_lanes` instead, to specify the exact ILKN lanes' bitmap)
- `Ilkn_interface_status_ignore`
- `Ilkn_interface_status_oob_ignore`
- `Ilkn_invalid_lane_id`
- `ilkn_tdm_dedicated_queuing` (use the `BCM_PORT_ADD_EGRESS_INTERLEAVING_ENABLE` flag in the `bcm_port_add` API for the master port, or use the `egr_ilv` extension for the `ucode_port` SOC property.

NOTE: The `Ilkn_is_burst_interleaving` SOC property is not relevant for the BCM88690 but is used in the BCM88800 and BCM88480 devices.

5.16.3 APIs

5.16.3.1 Changed APIs

The `bcm_port_add` API has changed. Before the BCM88800 and BCM88480 devices, the value of `phy_pbmp` (a bitmap of PHY lanes) for ILKN ports was a 24-bit bitmap indicating the active lanes on the ILKN core.

Starting with the BCM88800 and BCM88480, the `phy_pbmp` value is a bitmap of the device lanes (NIF or fabric), as defined in the `lane_to_serdes_map` SOC property described in the previous section.

After adding an ILKN port using interface lanes, configure the lane order by using the `bcm_port_ilkn_lane_map_set` API.

5.16.3.2 New APIs

The `bcm_port_ilkn_lane_map_set` and `bcm_port_ilkn_lane_map_get` APIs have been added.

These APIs configure the order of the ILKN lanes.

Each index in the `lanes[]` array represents an ILKN lane ID. The values of the `lanes[]` array are the NIF or fabric lanes mapped to ILKN logical lanes.

5.17 Linkscan

5.17.1 API

For `bcm_linkscan_handler_t`, only `linkstatus` is updated in the `info(bcm_port_info_t)` structure.

5.18 Diagnostic

The following diagnostics were changed during the upgrade from BCM88670 to BCM88690**:**

- `diag nif` was renamed to `nif status`.
- `diag port` was changed to `port mgmt.dump`. Also, the output format was modified, though the same information will be presented.
- `phy` diagnostic was changed to support the new diagnostic command format.

5.19 External PHY

An external PHY is not supported as an integrated part of the BCM88690 driver.

NOTE: The MDIO access APIs `bcm_port_phy_set` and `bcm_port_phy_get` are still valid for MDIO connection with an external PHY.

Chapter 6: DRAM and HBM

DRAM tune SOC properties have changed. To learn how to generate tuning properties, refer to the *Traffic Manager Programming Guide* (88690-PG2xx).

Chapter 7: Data Interfaces and Ports

7.1 APIs

`bcm_stk_sysport_gport_set` and `bcm_stk_sysport_gport_get`

Setting or getting the mapping of system port to modport is only supported with the system port gport.

Also, there is no need to create system ports for FMQ classes. These APIs no longer support `BCM_COSQ_GPORT_IS_FMQ_CLASS` gports.

Chapter 8: Trunk (Link Aggregation) Ports

8.1 Port Selection Criteria

Round Robin

This port selection criteria (PSC) is not supported by the BCM88690.

8.2 APIs

`bcm_trunk_find`

- The interface for this function was not changed; however, the valid input was changed.
- The input used to include unit, local port, and modid.
- The new input format should be unit and system port gport in the port parameter, and the modid parameter is ignored.

```
bcm_switch_control_port_set(unit, memberPort, bcmSwitchMcastTrunkHashMin, min);
bcm_switch_control_port_set(unit, memberPort, bcmSwitchMcastTrunkHashMax, max);
```

These APIs are no longer in use. In previous generations, their usage was optional to manually set pruning ranges for trunk members when egress MC is used. In the current generation of devices, the distribution and resolution of trunks in egress MC groups is done by a new mechanism. For more information, refer to the *Traffic Manager Programming Guide* (88690-PG2xx).

`bcm_trunk_create`

The trunk ID must be created using the macro `BCM_TRUNK_ID_SET(trunk_id, pool, group)`.

8.3 Member Flags

The `BCM_TRUNK_MEMBER_INGRESS_DISABLE` flag is no longer valid in the BCM88690. To achieve the same effect, in the relevant APIs, use the flag `BCM_TRUNK_MEMBER_EGRESS_DISABLE` instead.

8.4 Graceful LAG Modification

Graceful LAG modification is not supported in the BCM88690 and later devices.

The `system_ftmh_load_balancing_ext_mode = STANDBY_MC_LB` SOC property is not supported by the BCM88690.

Chapter 9: Ingress Traffic Management

9.1 Main Changes

Ingress Shaping Queue (ISQ) is not supported by the BCM88690.

9.2 Queues Creation

9.2.1 APIs

`bcm_cosq_ingress_queue_bundle_gport_add()` and `bcm_cosq_gport_delete()`

- ISQ is not supported by the BCM88690 (flag `BCM_COSQ_GPORT_ISQ`).
- The software-only option (flag `BCM_COSQ_GPORT_SW_ONLY`) is not supported.
- The API requires sysport gport as input (does not support modport as in the BCM88670).
- The core must be set to `BCM_CORE_ALL`.
- When creating FMQs in enhanced MC scheduling mode, the provided system port is ignored (because it is irrelevant).
- It is no longer necessary to create system ports for FMQ classes. `bcm_stk_sysport_gport_set()` does not support `BCM_COSQ_GPORT_IS_FMQ_CLASS` gports.

`bcm_cosq_gport_add()` and `bcm_cosq_gport_delete()`

- ISQ is not supported by the BCM88690 (flag `BCM_COSQ_GPORT_ISQ`).
- The API automatically assigns the queue to a credit-request profile. Therefore, this credit-request profile must be created before using the API.

  The following cint reference demonstrates how to create those profiles:

  `SDK/src/examples/dnx/ingress_tm/cint_credit_request_profile.c`
- The API requires sysport gport as input (does not support modport as in the BCM88670).
- When creating FMQs in an enhanced MC scheduling mode, the provided system port is ignored (it is irrelevant).
- It is longer needed to create system ports for FMQ classes. `bcm_stk_sysport_gport_set()` does not support `BCM_COSQ_GPORT_IS_FMQ_CLASS` gports.

`bcm_fabric_control_set()` and `bcm_fabric_control_get()`

- The BCM88670 supports three different queues (multicast queues, ISQs, and standard queues). Set queue type ranges to define queue types by using six control types: `bcmFabricQueueMin`, `bcmFabricQueueMax`, `bcmFabricShaperQueueMin`, `bcmFabricShaperQueueMax`, `bcmFabricMulticastQueueMin`, `bcmFabricMulticastQueueMax`.
- The BCM88690 supports two types of queues, which can be defined by a single queue number threshold (`bcmFabricMulticastQueueMax`). The other control types are not supported.

9.2.2 SOC Properties

`flow_mapping_queue_base`

- This SOC property is not supported by the BCM88690.
- The packet flow-id is equal to the base queue ID. Setting a custom mapping is not supported.

`hqos_mapping_enable`

- This SOC property is not supported by the BCM88690.
- `hqos_mapping_enable` was used to allow mapping of several system ports to a single module port.
  The BCM88690 SDK allows it in any mode (no need to configure this SOC property).

`voq_mapping_mode`

- Only INDIRECT mode is supported.

9.3 Queue Scheduling

9.3.1 APIs

9.3.1.1 Credit Worth

`bcm_fabric_control_set()` and `bcm_fabric_control_get()`

- `bcmFabricCreditSize` – The credit worth of a local device cannot change dynamically.
  The local credit size should be configured using the SOC property `credit_size`.
- `bcmFabricCreditSizeRemoteDefault` – The credit worth of remote egress devices should be configured using the API `bcm_cosq_dest_credit_size_set()`.

9.3.1.2 Credit Request Profiles

`bcm_cosq_delay_tolerance_level_set()` and `bcm_cosq_delay_tolerance_level_get()`/ `bcm_cosq_ingress_queue_bundle_gport_add()`

- All 32 credit-request profiles (`delay_tolerance_level`) are controlled by the user. Fixed profiles (delay tolerance level) are no longer created by default.
- To get recommended defaults, use the `bcm_cosq_delay_tolerance_preset_get()` API.
- The following cint reference demonstrates how to create legacy profiles to reuse legacy application code:

  `SDK/src/examples/dnx/ingress_tm/cint_credit_request_profile.c`, in the function `cint_credit_request_profile_backward_compatibilty_set()`.
- If the legacy API `bcm_cosq_gport_add()` is used by the application to create VOQs, the cint reference must be used (the cint creates the default profile that is used by `bcm_cosq_gport_add()`).

9.3.2 BCM88670 to BCM88800 and BCM88480 Compatibility

9.3.2.1 Priority per Queue

In the BCM88800 and BCM88480, the priority is determined per queue instead of per credit request profile as in the BCM88670 generation of devices.

In addition, in BCM88800, there are eight priorities instead of two, as in the BCM88690. Use `bcm_cosq_control_set()` and `bcm_cosq_control_get()` instead of the flag `BCM_COSQ_DELAY_TOLERANCE_HIGH_Q_PRIORITY` in `bcm_cosq_delay_tolerance_level_set`.

Syntax

`bcm_cosq_control_set()` and `bcm_cosq_control_get()`

Parameters

IN	port	VOQ Gport ■ `BCM_GPORT_UCAST_QUEUE_GROUP` ■ `BCM_GPORT_MCAST_QUEUE_GROUP`
IN	cosq	Queue offset from base queue
IN	type	`bcmCosqControlPrioritySelect`
IN/OUT	arg	Priority (0 to 7)

Syntax

```
cm_cosq_control_set (int unit, bcm_gport_t port, bcm_cos_queue_t cosq,
   bcm_cosq_control_t type, int arg)
```

Parameters

IN	port	Not relevant, should be 0.
IN	cosq	Not relevant, should be –1.
IN	type	`bcmCosqControlIngressMaxLowPriority`.
IN/OUT	arg	The maximal priority which is Low. Range: 0 to 7.

9.3.2.2 Zero Rate on FMQ Shaper

In the BCM88800, zero cannot be provided as the bandwidth (`kbits_sec_max`) for best-effort and guaranteed FMQ shapers.

9.4 Ingress Congestion Management

9.4.1 APIs

9.4.1.1 Implicit Management of VOQ Rate Class

The BCM88670 supports setting VOQ rate-class thresholds, either per-rate class (explicitly) or per queue (implicitly). The BCM88690 does not support implicit setting of the VOQ rate-class thresholds, and all VOQ rate-class thresholds can only be set using rate-class gport.

The cosq parameter is not relevant with rate-class gport. Rate-class threshold APIs validate that the cosq parameter is either 0 or –1. (In previous devices, it was not tested.)

9.4.1.2 VOQ Rate Class Creation

The BCM88690 introduces a new API to create rate classes. This API sets the default threshold for the rate class according to the requested rate and additional attributes.

Every rate class used should first be created using the designated API.

Syntax

```
bcm_cosq_gport_rate_class_create(int unit, bcm_gport_t gport, uint32 flags,
bcm_cosq_rate_class_info_t *create_info);
```

Parameters

I		
N	gport	Rate class gport
IN	flags	0
IN	create_info. rate	Approximately the desired rate in Gb/s.

IN	create_info. attributes	`BCM_COSQ_RATE_CLASS_CREATE_ATTR_SLOW_ENABLED`
		`BCM_COSQ_RATE_CLASS_CREATE_ATTR_OCB_ONLY`
		`BCM_COSQ_RATE_CLASS_CREATE_ATTR_MULTICAST`

9.4.1.3 Tail Drop/FADT

FADT is always enabled in the BCM88690, and it is the only way to configure a drop.

- `bcm_cosq_control_set()` and `bcm_cosq_control_get()`.
- `bcmCosqControlDropLimitAlpha` type is deprecated.
- `bcmCosqControlOcbFadtDropEnable` type is deprecated.

`bcm_cosq_gport_color_size_set()` and `bcm_cosq_gport_color_size_get()`

- `BCM_COSQ_GPORT_SIZE_BUFFER_DESC` flag is deprecated, both for VOQ and VSQ.
- Tail drop is always FADT.
- `size.size_min` with the `BCM_COSQ_GPORT_SIZE_BUFFERS` and `BCM_COSQ_GPORT_SIZE_OCB` flags set configures guaranteed OCB buffers and not a static-free resource threshold as in the BCM88670.

9.4.1.4 VOQ Categories

The BCM88690 supports only two VOQ categories: multicast queue (category 0) and unicast queue (category 2).

Partitioning of VOQ into these categories is done automatically by the `bcm_fabric_control_set()` API, which defines the range of FMQ. The following APIs are deprecated:

`bcm_cosq_control_range_set()` and `bcm_cosq_control_range_get()`

- `bcmCosqRangeMulticastQueue` type is deprecated
- `bcmCosqRangeS` type is deprecated
- `haperQueue` type is deprecated
- `bcmCosqRangeFabricQueue` type is deprecated
- `bcmCosqRangeRecycleQueue` type is deprecated

`bcm_fabric_control_set()` and `bcm_fabric_control_get()`

- `bcmFabricVsqCategory` type is deprecated

9.4.1.4.1 Default Invalid Queue

BCM88690 supports mapping to a default invalid queue. This queue is used to count all packets sent to invalid destinations. Traffic is not forwarded to this queue. The API remains the same.

`bcm_cosq_control_set()` and `bcm_cosq_control_get()`

- `bcmCosqControlDefaultInvalidQueuetype`
- The only supported gports are:
  - `BCM_GPORT_UNICAST_QUEUE_GROUP_SET`
  - `BCM_GPORT_MCAST_QUEUE_GROUP_SET`

9.4.1.5 Global Discard per DP

DP3 is always dropped.

The `bcm_cosq_discard_set` API does not support flags = 0.

At the ingress side, supported flags are as follows:

- `BCM_COSQ_DISCARD_ENABLE | BCM_COSQ_DISCARD_COLOR_BLACK` – Discards packets with DP = 3.
- `BCM_COSQ_DISCARD_ENABLE | BCM_COSQ_DISCARD_COLOR_BLACK | BCM_COSQ_DISCARD_COLOR_RED` – Discards packets with DP ≥ 2.
- `BCM_COSQ_DISCARD_ENABLE | BCM_COSQ_DISCARD_COLOR_BLACK | BCM_COSQ_DISCARD_COLOR_RED | BCM_COSQ_DISCARD_COLOR_YELLOW` – Discards packets with DP ≥ 1.
- `BCM_COSQ_DISCARD_ENABLE | BCM_COSQ_DISCARD_COLOR_ALL` – Discards all packets.

9.4.1.6 Global VSQ Threshold

The BCM88690 has a new API to set VSQ global thresholds.

`bcm_cosq_gport_threshold_set()` and `bcm_cosq_gport_threshold_get()`

- `BCM_COSQ_THRESHOLD_INGRESS` flag with global VSQ gport is deprecated.
- Use `Bcm_cosq_gport_static_threshold_set`.

9.4.1.7 VSQ Resource Allocation

`bcm_cosq_resource_allocation_set()` and `bcm_cosq_resource_allocation_get()`

- `bcmReservationResourceBufferDescriptors` resource is deprecated.
- `target.is_ocb_only` is ignored.

9.4.1.8 OCB Eligibility

The BCM88690 does not have an OCB-eligibility mechanism.

`bcm_cosq_gport_threshold_set()` and `bcm_cosq_gport_threshold_get()`

The `BCM_COSQ_THRESHOLD_INGRESS` and `BCM_COSQ_THRESHOLD_OCB` flag combination is deprecated.

9.4.1.9 Explicit Congestion Notification (ECN)

ECN is always enabled in the BCM88690. The following API is deprecated:

`bcm_cosq_discard_set()` and `bcm_cosq_discard_get()`

The `BCM_COSQ_DISCARD_MARK_CONGESTION` flag is deprecated.

9.4.1.10 Action Signature

The BCM88690 does not have action signature mechanism.

`bcm_cosq_control_set()` and `bcm_cosq_control_get()`

The `bcmCosqControlHeaderUpdateField` type is deprecated.

9.4.1.11 Admission Profile

The BCM88690 does not have admission profile mechanism.

`bcm_cosq_control_set()` and `bcm_cosq_control_get()`

The `bcmCosqControlAdmissionTestProfileA` and `bcmCosqControlAdmissionTestProfileB` types are deprecated.

9.4.2 SOC Properties

`ingress_congestion_management_guarantee_mode`

- SOC property is not supported by the BCM88690.
- Guaranteed resources are managed independently for VOQ and VSQ, similar to the LOOSE mode in the BCM88670.

`ingress_congestion_management_stag_max_id`

The SOC property is not supported by the BCM88690.

`ingress_congestion_management_tm_port_max_id`

The SOC property is not supported by the BCM88690.

cosq_admission_preference

The SOC property is not supported for DP3. For DP3, it is always set to `ADMIT_OVER_GUARANTEE`.

9.5 Priority Drop

The BCM88690 priority drop (PRD) mechanism resembles the BCM88680 priority drop mechanism, which is substantially different from the BCM88670. This section summarizes the difference between the BCM88690 and BCM88680 devices.

The BCM88690 has a PRD parser unit per octet (eight NIF lanes), but the BCM88680 has a PRD parser unit per quad (four NIF lanes). All APIs that affect an entire quad in the BCM88680 will affect an entire octet in the BCM88690 unless stated otherwise.

9.5.1 APIs

9.5.1.1 bcm_dnx_cosq_ingress_port_drop_enable_set() and bcm_dnx_cosq_ingress_port_drop_enable_get()

```
bcm_cosq_ingress_port_drop_enable_set(int unit, bcm_port_t port, uint32 flags, int
enable_mode);
```

BCM88690 does not support HiGig™ mode.

9.5.1.2 bcm_cosq_ingress_port_drop_threshold_set() and bcm_cosq_ingress_port_drop_threshold_get()

```
bcm_cosq_ingress_port_drop_threshold_set(int unit, bcm_port_t port, uint32 flags,
int priority, uint32 value);
```

- For the BCM88680, this API affects the entire quad. The maximum threshold value is derived from the quad mode (single, dual, and quad mode).
- For the BCM88690, this API interacts with a new API, `bcm_port_priority_config_set`. For more information, refer to the Hardware Interfaces section in *Traffic Manager Programming Guide* (88690-PG2xx). The threshold configuration is per priority group (RMC). The maximum threshold value is the maximum FIFO size, which is configurable.
- The BCM88680 has three configurable thresholds. (The threshold for priority 3 equals the FIFO size.) The BCM88690 has four configurable thresholds, one per priority.

9.5.1.3 bcm_cosq_ingress_port_drop_flex_key_construct_set() and bcm_cosq_ingress_port_drop_flex_key_construct_get()

```
bcm_cosq_ingress_port_drop_flex_key_construct_set(int unit,
bcm_cosq_ingress_drop_flex_key_construct_id_t *key_id, uint32 flags,
bcm_cosq_ingress_drop_flex_key_construct_t *flex_key_config);
```

- For the BCM88680, key construction is common to all EtherTypes. For the BCM88690, the key to the TCAM is constructed per EtherType. That is, the key to the TCAM can be constructed differently for each EtherType.
- To support this feature, the following new field was added to `key_id`:

  `key_id.ether_type_code` – Configured EtherType code.
- For the BCM88680, the offset base is at the start of the packet. For the BCM88690, the offset base is configurable.

  To support this feature, the following two new fields were added to `flex_key_config`:
  - `flex_key_config.offset_base` – Offset base mode.
  - `flex_key_config.ether_type_header_size` – The header size that comes after the Ethernet header.

  For compatibility with the BCM88680, make sure to initialize `flex_key_config` with the following API:

  `bcm_cosq_ingress_drop_flex_key_construct_t_init`.

9.6 Ingress Shapers

9.6.1 APIs

9.6.1.1 bcm_switch_control_set() and bcm_switch_control_get()

```
bcm_switch_control_set(int unit, bcm_switch_control_t type, int arg)
```

In legacy devices, calling with the `bcmSwitchIngressRateLimitMpps` type and `arg = 0` unsets the limit. In the BCM88690, this call limits the rate to 0 Mp/s.

Chapter 10: Multicast

10.1 Deprecated APIs

Multicast legacy APIs that support only one encapsulation ID are no longer supported.

Instead, call the following new set of APIs:

- `bcm_multicast_add()`
- `bcm_multicast_delete()`
- `bcm_multicast_set()`
- `bcm_multicast_get()`

The `delete_all` APIs are also not supported. The functionality is supported by the `bcm_multicast_delete()` API called with the `BCM_MULTICAST_REMOVE_ALL` flag.

The `bcm_multicast_init()` and `bcm_multicast_detach()` APIs are no longer needed. The multicast module is initialized as a part of `bcm_init()`.

10.1.1 Unsupported APIs

- `bcm_multicast_ingress_add()`, `bcm_multicast_egress_add()`, `bcm_multicast_ingress_delete()`, and `bcm_multicast_egress_delete()`
- `bcm_multicast_ingress_delete_all()` and `bcm_multicast_egress_delete_all()`
- `bcm_multicast_ingress_get()` and `bcm_multicast_egress_get()`
- `bcm_multicast_ingress_set()` and `bcm_multicast_egress_set()`
- `bcm_multicast_init()`
- `bcm_multicast_detach()`

10.2 Deprecated Flags

The flags described in this section, which are used with the `bcm_multicast_*()` APIs, are deprecated and are not supported. These flags were not processed by the API in legacy devices. For the BCM88690, calling the API with these flags is not necessary, and the flags should be removed.

10.2.1 Unsupported Flags

- `BCM_MULTICAST_TYPE_L2`
- `BCM_MULTICAST_TYPE_L3`
- `BCM_MULTICAST_TYPE_VPLS`
- `BCM_MULTICAST_TYPE_TRILL`
- `BCM_MULTICAST_TYPE_MIM`
- `BCM_MULTICAST_TYPE_EGRESS_OBJECT`
- `BCM_MULTICAST_TYPE_SUBPORT`

10.2.2 bcm_multicast_add(), bcm_multicast_delete(), bcm_multicast_set(), and bcm_multicast_get() Flags

`BCM_MULTICAST_INGRESS` is not supported.

In the legacy devices, the egress flag was unnecessary. If the ingress flag was not mentioned, the egress was referenced.

In the BCM88690 (Jericho2) device, ingress or egress must be specified using the new flags with the `bcm_multicast_add()`, `bcm_multicast_delete()`, `bcm_multicast_set()`, and `bcm_multicast_get()` APIs.

10.2.2.1 New Flags

- `BCM_MULTICAST_EGRESS_GROUP`
- `BCM_MULTICAST_INGRESS_GROUP`

10.2.2.2 Unsupported Flag

`BCM_MULTICAST_INGRESS`

10.2.3 bcm_fabric_multicast_set()

Because `SINGLE_FAP` mode is treated as mesh, call this API when configuring egress multicast over a device in `SINGLE_FAP` mode.

10.2.4 Trunk gport

In BCM88690 (Jericho2), the `bcm_multicast_get()` API returns trunk gport when the destination is LAG gport.

On legacy devices, this API returns trunk members instead of the trunk gport itself.

10.3 CUD Restrictions

CUD only (no destination port) and double CUD (specifying two CUDs per replication) are not supported for `bcm_multicast_add()`, `bcm_multicast_delete()`, `bcm_multicast_set()`, and `bcm_multicast_get()`.

10.3.1 Unsupported Options

- CUD only – port = `BCM_GPORT_INVALID`
- Double CUD – encap2 != `BCM_IF_INVALID`

10.4 Direct Bitmap

Direct bitmap is not supported.

10.4.1 Unsupported SOC Properties

- `egress_multicast_direct_bitmap_max`
- `egress_multicast_direct_bitmap_min`

10.5 Adding Multiple Replications

In legacy devices, multicast APIs prevented adding the same replications to a multicast group more than once. The BCM88690 device does not check this scenario. It is the responsibility of the user to avoid it.

10.5.1 Affected APIs

`bcm_multicast_add()` and `bcm_multicast_set()`

10.6 Deleting a Non-Existing Replication

In legacy devices, multicast APIs return an error when deleting a nonexisting replication. BCM88690 does not return an error in this case. If a replication does not exist, it is not deleted.

10.6.1 Affected APIs

`bcm_multicast_delete()`

10.7 Fabric Multicast

The BCM88480 and BCM88280 devices do not support fabric multicast. Instead of using fabric multicast with an egress multicast scheme, use ingress an ingress multicast and egress multicast scheme or ingress multicast only.

Chapter 11: Sniff

11.1 Sequence Change

In the BCM88670, the forwarding port was assigned with a recycle mirror port in API. `bcm_mirror_port_vlan_destination_add()` with flag `BCM_MIRROR_PORT_EGRESS_ACL`.

The recycle mirror port allocation and mapping from the forwarding port to it was implicitly done by the SDK.

In the BCM88690, the `bcm_mirror_port_vlan_destination_add()` API with the `BCM_MIRROR_PORT_EGRESS_ACL` flag is deprecated.

To allocate and map a recycle mirror port, the following sequence is required:

1. Create the recycle mirror port using the `bcm_port_add` API with interface `BCM_PORT_IF_RCY_MIRROR`.
2. Map the forwarding port to the recycle mirror port using the `bcm_mirror_port_to_rcy_port_map_set()` API.

For additional details, refer to the Sniff section in the *Traffic Manager Programming Guide* (88690-PG2xx).

11.2 Deprecated Controls

The following control, used with the `bcm_switch_control_set()` API, is deprecated and not supported. This functionality is supported per sniff profile and added to the `bcm_mirror_destination_create()` API.

11.2.1 Unsupported Control

`bcmSwitchMirrorSnoopForwardOriginalWhenDropped`

11.3 Deprecated Flags

The following flags, used with the `bcm_mirror_destination_create()` API, are deprecated and are not supported. For counter functionality, a new API is added:

- `bcm_mirror_destination_count_cmd_set()`

11.3.1 Unsupported Flags

- `BCM_MIRROR_DEST_UPDATE_COUNTER`
- `BCM_MIRROR_DEST_UPDATE_COUNTER_1`
- `BCM_MIRROR_DEST_UPDATE_COUNTER_2`
- `BCM_MIRROR_DEST_EGRESS_TRAP_WITH_SYSTEM_HEADER`

Chapter 12: Ingress Compensation

The BCM88670 has two compensation modes: per-queue compensation (default) and per-packet compensation.

The BCM88690 has one compensation mode: per-packet compensation. In addition, compensation values are common between different compensation types.

12.1 APIs

12.1.1 Per-Queue Compensation

`bcm_cosq_control_set()` and `bcm_cosq_control_get()`

- `bcmCosqControlPacketLengthAdjust` is not supported for the VOQ gport.

`bcm_cosq_pkt_size_adjust_delta_map_set()` and `bcm_cosq_pkt_size_adjust_delta_map_get()` are unsupported APIs.

12.1.2 Per-Packet Compensation

`bcm_cosq_gport_pkt_size_adjust_set()` and `bcm_cosq_gport_pkt_size_adjust_get()`

1. The following types are not supported:
   - `bcmCosqPktSizeAdjustSourceScheduler`
   - `bcmCosqPktSizeAdjustSourceCrpsInPP`
   - `bcmCosqPktSizeAdjustSourceCrpsInTM`
   - `bcmCosqPktSizeAdjustSourceStatReportIn`
   - `bcmCosqPktSizeAdjustSourceStatReportOut`

   `bcmCosqPktSizeAdjustSourceGlobal` should be used for all compensation types
2. In Scheduler compensation, the per-input port-compensation component is disabled by default. (It is enabled in the BCM88670.) The API to enable per-input port-compensation component is provided.
3. In Counter Processor and Statistic Interface compensation, for packet size adjust using IRPP/ITM mask by the default Header Truncate is disabled. (It is configurable in the BCM88670 using the SOC property.)

   The API to enable and disable header-truncate compensation is provided (`bcm_stat_pkt_size_adjust_select_set` and `bcm_stat_pkt_size_adjust_select_get`).

12.2 SOC Properties

`ingress_congestion_management_pkt_header_compensation_enable`

- Unsupported SOC property.

`truncate_delta_in_pp_counter`

- Unsupported SOC property; use the `bcm_stat_pkt_size_adjust_select_set()` API instead.

`truncate_delta_in_tm_counter`

- Unsupported SOC property, use the `bcm_stat_pkt_size_adjust_select_set()` API instead.

Chapter 13: Fabric Data Path

13.1 Packet Cell Packing

Packet cell packing (PCP) includes the following characteristics:

- PCP can be configured on MESH.
- PCP can be configured for the local route.

13.1.1 SOC Properties

`Fabric_pcp_enable`

- The SOC property is not supported in the BCM88690.
- PCP is enabled by default, which means that VSC256_v2 will always be used (allowed because there is no direct connection to the FE1600).

13.2 Fabric Priority

13.2.1 API

`bcm_fabric_priority_set()` and `bcm_fabric_priority_get()`

The flag `BCM_FABRIC_PRIORITY_TDM` is supported only on the BCM88800 and BCM88480.

13.2.2 SOC Properties

`fabric_tdm_priority_min`

- In the BCM88690 this value is ignored because there is no TDM.
- In the BCM88800 and BCM88480, only the following values are supported:
  - 3
  - None

13.3 Mesh

13.3.1 API

`bcm_fabric_static_replication_set`

This API is not supported. Use the `bcm_fabric_multicast_set()` API instead.

13.3.2 SOC Properties

`fabric_mesh_multicast_enable`

- The SOC property accepts only the values 0 (TDM-MC only) or 1 (full MC). The value 2 (TDM MC using IRE) is not supported in the BCM88690 and later devices.

  NOTE: In the BCM88670, mode 0 did not support MC, while in the BCM88690 and later devices, it is supported.
- In mesh mode, UC and MC traffic cannot use the same pipe. Therefore, when multicast is enabled (`fabric_mesh_multicast_enable=1`), the UC/MC 2-pipes must also be configured.

`fabric_connect_mode`

When setting the SOC property to `SINGLE_FAP`, the device is configured the same way it is in mesh mode, with fabric SerDes disabled and without software loopbacks over the fabric links.

13.4 Fabric Connected to a Repeater

The BCM88690 and later devices are not expected to work directly versus a repeater.

13.4.1 API

`bcm_fabric_link_control_set()` and `bcm_fabric_link_control_get()`

The type `bcmFabricLinkRepeaterEnable` is not supported.

13.4.2 SOC Properties

`repeater_link_enable_<link-id>=<0/1>`

The SOC property is not supported in the BCM88690 and later devices.

13.5 Fabric Interface FIFOs

The Fabric WFQs changed for the BCM88690. For details, refer to the *Traffic Management Architecture* design guide (for the BCM88690.

13.5.1 API

`bcm_cosq_gport_sched_set()` and `bcm_cosq_gport_sched_get()`

Configure Egress (FDR) WFQ using `bcmCosqGportTypeFabricPipeEgress` changed. The configuration is now done using `bcm_cosq_gport_dynamic_sched_set()` and `bcm_cosq_gport_dynamic_sched_get()`. Refer to the *Traffic Manager Programming Guide* for details.

13.6 Fabric Data Cell to Core Mapping

13.6.1 SOC Properties

`fabric_links_to_core_mapping_mode=SHARED`

- The BCM88690 and later devices support only SHARED mode.
- DEDICATED mode is not supported.
- SHARED mode is the default mode, so the SOC property is redundant and is being kept only for backward compatibility.

13.7 Fabric Congestion Management

- In the BCM88670, fabric congestion management (including RCI, GCI, LLFC, and Drop) was configured through various APIs.
- In the BCM88690 and later devices, all fabric congestion configurations are done through a single API, `bcm_fabric_cgm_control_set()`.
- Because the GCI mechanism has not changed in the BCM88690 (and later devices), these new devices continue to support legacy BCM88670 APIs that configure GCI. However, it is best practice to use the new APIs.

The following RCI controls are not supported for single-core devices:

- `bcmFabricCgmRciEgressPipeLevelTh`
- `bcmFabricCgmRciTotalEgressPipeLevelTh`
- `bcmFabricCgmRciTotalLocalLevelTh`
- `bcmFabricCgmRciEgressLevelFactor`
- `bcmFabricCgmRciCoreLevelMappingTh`

13.8 Fabric Logical Port Base

13.8.1 SOC Properties

The default values for the `fabric_logical_port_base` SOC property are as follows:

- For the BCM88690, `fabric_logical_port_base` is 256 by default.
- For the BCM88800, `fabric_logical_port_base` is 1024 by default.

Chapter 14: Egress Traffic Management

14.1 Recycle Interface

In the BCM88690, there are two recycle interfaces per core instead of one. When the recycle port is created, it should be indicated to which interface it belongs (0 or 1).

14.1.1 API

`bcm_port_add()` and `bcm_port_get()`

`interface_info.interface_id = 0/1;`

14.2 Priority Propagation

Propagation priority (high or low) is now configured per port-priority (queue-pair) instead of per port.

14.2.1 API

`bcm_cosq_control_set()` and `bcm_cosq_control_get()`

- `port` should be of type `BCM_COSQ_GPORT_PORT_TC_SET`.
- `cosq` should be the priority.

NOTE: Calling the API, as in the BCM88670 device, with a port of type `BCM_GPORT_LOCAL_SET` will also work and set all the port priorities.

14.3 Header Compensation

The header compensation mechanism was changed, and only the network header size should be indicated when configuring egress compensation. The rest of the packet size changes (PP editing, system headers, and so on) are accounted for internally by hardware. For more information, refer to the *Traffic Manager Programming Guide*, the Header Compensation subsection in the Egress Traffic Management chapter.

14.3.1 API

`bcm_cosq_control_set()` and `bcm_cosq_control_get()`

- `bcmCosqControlPacketLengthAdjust` is not supported for egress multicast queue gport.

14.4 Single Calendar Mode

Single calendar mode, in which the calendar shaper becomes irrelevant and only interface shaper actually works, is no longer supported.

14.4.1 API

`bcm_cosq_control_set()` and `bcm_cosq_control_get()`

`bcmCosqControlSingleCalendarMode` is not supported.

14.5 Egress Congestion Management

14.5.1 Warnings and Errors

As a general note, all the supported functionality for the ECGM API is described in the *Traffic Manager Programming Guide*. Any functionality not mentioned in this document should be treated as unsupported.

In previous SDK versions, some parameters and values were not verified by the ECGM API, and errors were not returned correctly.

NOTE: Some previously functioning usage may now result in an error (or a warning). To avoid any dysfunction, consult the *Traffic Manager Programming Guide*.

14.5.2 SOC Properties

The following SOC properties are not supported in the BCM88690:

- `egress_fabric_drop_threshold_multicast_low`
- `egress_fabric_drop_threshold_multicast`
- `egress_fabric_drop_threshold_all_except_local_tdm`
- `egress_fabric_drop_threshold_all`
- `egress_shared_resources_mode`

Shared resources mode is set to strict as default.

ECGM thresholds may be modified dynamically using the API. For details, review the Egress Traffic Management chapter in the *Traffic Manager Programming Guide*.

14.6 Shaping Mode Selection

On a logical-port level in the previous generation of devices, shaping can be done in a byte mode or a packet mode. This is done by using SOC property that is not supported in the current generation of devices.

In the BCM88690, the shaping mode is selected per interface and is done by using an API. For more information, refer to the *Traffic Manager Programming Guide* (`bcm_cosq_control_set` with `bcmCosqControlShaperPacketMode` type).

14.6.1 SOC Property

The following SOC property is not supported in the BCM88690:

```
otm_port_packet_rate
```

14.6.2 Service Pools

14.6.2.1 Mapping Resources Service Pools

In previous generations, mapping to both packet descriptor (PD) and data buffer (DBs) service pools was done per multicast TC (MC-TC).

In the BCM88690, MC-TC is mapped to DB service pools, and multicast queues are mapped to PD service pools.

Mapping a PD service pool to a multicast queue is performed by using an API, as described in the following section.

14.6.3 bcm_cosq_control_set

Used with the type `bcmCosqControlMulticastQueueToPdSpMap`.

Mapping DB service pool to a MC-TC is done, as in previous generations, using the following API:

```
bcm_cosq_gport_egress_multicast_config_set
```

For details, review the *Traffic Manager Programming Guide*, Egress Traffic Management chapter.

14.6.3.1 Modifying DB Service Pool Mapping

To change service pool mapping for an MC-TC, set the reserved DB threshold to 0.

14.6.3.2 Service Pool Reserved Thresholds

14.6.3.2.1 APIs

`bcm_cosq_gport_threshold_set/bcm_cosq_gport_threshold_get`

- Setting total reserved thresholds per PD/DB service pool is not supported in the BCM88690.
- The reserved value can only be modified per queue (for reserved PDs) or per MC-TC (for reserved DBs).
- Using the API with the parameters in the following table will return an error.

Gport	Cosq	Flags	Type
Core	-1	`BCM_COSQ_THRESHOLD_EGRESS` | `BCM_COSQ_THRESHOLD_MULTICAST_SP0/1`	`bcmCosqThresholdAvailablePacketDescriptors`
Core	-1	`BCM_COSQ_THRESHOLD_EGRESS` | `BCM_COSQ_THRESHOLD_MULTICAST_SP0/1`	`bcmCosqThresholdAvailableDataBuffers`

14.6.3.3 Multicast COS Parameters

14.6.3.3.1 APIs

`bcm_cosq_gport_egress_multicast_config_get` and `bcm_cosq_gport_egress_multicast_config_get`

- Used with flag: `BCM_COSQ_MULTICAST_UNSCHEDULED`
- The API no longer supports gport to indicate a single core.
- The configuration is performed at the device level (for both cores).
- Two (equivalent) options for valid gports are as follows:
  - core gport, set to `BCM_CORE_ALL`
  - gport == 0

For more information about changes to this API, see Section 14.6.3.1, Modifying DB Service Pool Mapping.

14.6.3.4 Interface Thresholds

14.6.3.4.1 APIs

`bcm_cosq_gport_threshold_set` and `bcm_cosq_gport_threshold_get`

Setting or getting thresholds per interface requires the threshold.priority indication as described in the *Traffic Manager Programming Guide*.

14.6.3.5 FADT for DB Service Pools

14.6.3.5.1 APIs

`bcm_cosq_gport_threshold_set` and `bcm_cosq_gport_threshold_get`

In the previous generation of devices, setting drop thresholds per TC for data buffers service pools had one Alpha factor for FADT. For better granularity, another Alpha factor was added in the BCM88690.

The Alpha factors are accessed using the types:

- `bcmCosqThresholdDataBuffersAlpha` (for Alpha 0)
- `bcmCosqThresholdDataBuffersAlpha1` (for Alpha 1)

To come as close as possible to the FADT behavior with one Alpha factor, the new Alpha 1 should be set to (–7). However, it will not be identical to the calculation of a single alpha.

For more information on FADT behavior, refer to the *Traffic Manager Programming Guide*.

Chapter 15: Egress (End-to-End) Credit Scheduler

15.1 APIs

`bcm_cosq_gport_attach()`

`bcm_port_resource_set()` must be performed before attaching a flow to port HR.

`bcm_cosq_control_set()/bcm_cosq_control_get()`

- `bcmCosqControlFlowSlowRate1`, `bcmCosqControlFlowSlowRate2`
  - Slow rate cannot be configured using `bcm_cosq_control_set`.
  - `bcm_cosq_slow_profile_set` should be used to configure slow rates.

The following controls are not supported by the BCM88690, and they did not perform any operation on the BCM88670 device.

- `bcmCosqControlFlowControlPriority`
- `bcmCosqControlSpLevelMax`
- `bcmCosqControlSp0WeightMax`
- `bcmCosqControlSp1WeightMax`
- `bcmCosqControlSp2WeightMax`
- `bcmCosqControlSp3WeightMax`
- `bcmCosqControlSp4WeightMax`
- `bcmCosqControlSp5WeightMax`
- `bcmCosqControlSp6WeightMax`
- `bcmCosqControlSp7WeightMax`
- `bcmCosqControlSp8WeightMax`
- `bcmCosqControlSp9WeightMax`
- `bcmCosqControlSp10WeightMax`

The following APIs are deprecated and replaced with `bcm_cosq_bandwidth_fabric_adjust_set()` and `bcm_cosq_bandwidth_fabric_adjust_get()`:

- `bcm_fabric_bandwidth_ profile_set()` and `bcm_fabric_bandwidth_ profile_get()`
- `bcm_fabric_bandwidth_core_profile_set()` and `bcm_fabric_bandwidth_core_profile_get()`

`bcm_cosq_gport_traverse`

The flags parameter is not updated by this API.

`bcm_cosq_gport_get`

- When called on a flow connected to port HR (either for TC = 0, or for TC != 0), the scheduler gport is returned.
- When called on port HR scheduler gport, the E2E port gport is returned.

`bcm_cosq_control_set` with `bcmCosqControlBandwidthBurstMax` control

`Max_burst` units for `E2E_PORT_TC` and `E2E_PORT_TCG` gports has changed. Provide the maximum burst in bytes.

15.2 SOC Properties

`slow_max_rate_level`

- SOC property is not supported by the BCM88690.
- There is no trade-off between the maximum slow rate and the maximum burst.

`runtime_performance_optimize_enable`

The SOC property is not supported by the BCM88690.

Flow Regions

`dtm_flow_mapping_mode_region`

- Interdigitated regions can be defined as type 1 only (InterDigitated = TRUE, OddEven = TRUE).
- Other types of interdigitated regions (2, 5, and 6) are not supported.
- HR regions' quartets are CL – HR – FQ – FQ on the BCM88690 (CL-HR-CL-FQ on BCM88800/BCM88480). No VOQ connectors exist in HR regions.

15.3 BCM88670 to BCM88800 and BCM88480 Compatibility

Flow Regions

`dtm_flow_mapping_mode_region`

The SE region quartet in the BCM88800 and BCM88480 devices is CL-FQ-CL-FQ, while in BCM88690, this quartet is CL-FQ-FQ-FQ. On all devices, SE type is the same: type 3/type 7.

`bcm_cosq_scheduler_gport_add`

Composite FQ is not supported on BCM88800 and BCM88480.

`bcm_dnx_cosq_bandwidth_fabric_adjust_set`

The `BCM_COSQ_BW_FABRIC_ADJUST_CONFIG_SHARED` flag is not supported on the BCM88800 or BCM88480.

Chapter 16: Flow-Control Configuration

16.1 Inband FC Configuration

Setting the FC mode of an interface with an SOC property is no longer supported.

This functionality is supported only with an API.

Unsupported SOC Property

`fc_inband_mode_<tx/rx><port>`

16.2 OOB FC Configuration

OOB FC HCFC and E2E are no longer supported.

COE is supported, but does not need to be configured with the SOC property.

Unsupported Values for SOC Property

`fc_oob_type<oob_id> = 0x3` for HCFC, `0x4` for COE, `0x5` for E2E.

Supported Values for SOC Property

`fc_oob_type<oob_id> = 0x1` for SPI, `0x2` for ILKN.

16.3 OOB Voltage Configuration

Setting the voltage mode for OOB interfaces is no longer supported.

Unsupported SOC Property

`ext_voltage_mode_oob`

16.4 ILKN Inband FC Configuration and Retransmit

ILKN inband FC and retransmit are not supported.

Unsupported SOC Properties

- `fc_inband_intlkn_`
- `ilkn_retransmit_`

16.5 HCFC Configuration

HCFC is not supported.

Unsupported types for bcm_cosq_control_set() API

- `bcmCosqControlFlowControlErrHandle`
- `bcmCosqControlFlowControlErrWatchdog`

16.6 E2E FC Configuration

E2E is not supported.

Unsupported SOC Properties

- `fc_calendar_e2e_status_nof_entries`
- `fc_calendar_indication_invert`

16.7 FC Path Concept

Some options are not supported for the FC path API:

```
int bcm_cosq_fc_path_add(
int unit,
bcm_cosq_fc_direction_type_t fc_direction,
bcm_cosq_fc_endpoint_t *source,
bcm_cosq_fc_endpoint_t *target);
```

Unsupported Reception Source Objects

- Inband calendar – `source` → `calender_index` != –1 && `source` → port == port gport
- MUB – `source` → `calender_index` != –1 && source → port == port gport

Unsupported Generation Source Objects

- HCFC bitmap – `source` → Flags: `BCM_COSQ_FC_HCFC_BITMAP`
- Global resource with `source` → Flags: `BCM_COSQ_FC_MINI_DB`, `BCM_COSQ_FC_FULL_DB`.
  - Other global resources are still supported.

Unsupported Generation Target Objects

- In-band calendar – `target` → `calender_index` != –1 && `target` → port == port gport
- MUB – `target` → `calender_index` != –1 && `target` → port == port gport

16.8 Threshold Configuration

The following options are not supported for the FC threshold API:

```
int bcm_cosq_gport_threshold_set(
int unit,
bcm_gport_t gport,
bcm_cos_queue_t cosq,
bcm_cosq_threshold_t *threshold);
```

The BCM88690 has new APIs to set and get FC thresholds for global resources:

`bcm_cosq_gport_threshold_set()` and `bcm_cosq_gport_threshold_get()`

Unsupported Threshold Types (Threshold → Type):

- `bcmCosqThresholdMiniDbuffs`
- `bcmCosqThresholdFullDbuffs`
- `bcmCosqThresholdDbuffs`
- `bcmCosqThresholdPacketDescriptorBuffers`
- `bcmCosqThresholdBufferDescriptors`
- `bcmCosqThresholdPacketDescriptor`

Unsupported Threshold Flags (Threshold → Flags):

```
BCM_COSQ_THRESHOLD_FLOW_CONTROL
```

Chapter 17: TDM and OTN Application

TDM is not supported in the BCM88690 device.

For other devices in the BCM88690 generation that support TDM, the APIs are not backward compatible with the APIs for the BCM88670 generation of devices. For details, refer to the *Traffic Manager Programming Guide*.

Chapter 18: Counter Processor

The counter processor implementation in the BCM88690 is significantly different from the BCM88670. The SDK was redesigned accordingly. Most of the APIs were changed, and the BCM88670 APIs are not applicable for the BCM88690.

For details, refer to the Counter Processor chapter in the *Traffic Manager Programming Guide*.

Chapter 19: Metering

The metering implementation in the BCM88690 is significantly different from the BCM88670.

For new SDK metering design details, refer to the Metering chapter in the *Traffic Manager Programming Guide*.

The main differences are as follows:

- In the BCM88670, Ethernet meter per port was made by dedicated API – `bcm_rate_bandwidth_set`.
- In the BCM88690, Ethernet meter per port is made as part of the generic database on database-id-0.
- In the BCM88670, `bcm_policer_create` was an independent API.
- In the BCM88690, primary to API `bcm_policer_create`, the user should create a generic meter database.
- Meter diagnostics for the BCM88690 are gathered under the key "meter".
- The assignment of PP objects to meters has changed. For information, refer to the *Packet Processing Programming Guide*.

Table 1: Main Modify and Removed APIs

BCM88670 API/Parameter	BCM88690 Change	Comments
`bcm_rate_bandwidth_set`	Removed	Limit traffic per port is made by ingress generic meter API (refer to the Flow section in the *Packet Processing Programming Guide*).
`bcm_policer_aggregate_group_create`	Removed. Similar API: `bcm_policer_expansion_groups_set`	Use `bcm_policer_expansion_groups_set` to create groups for L2-Fwd-Types.

Flags for APIs:
`bcm_policer_create`
`bcm_policer_set`
	Flag `BCM_POLICER_WITH_ID` removed – always with ID.
	Flag `BCM_POLICER_MACRO` removed.
		—

`bcm_policer_config_t` → `mark_drop_as_black`	Not in use. New parameter: `color_resolve_profile` which can be used as `mark_drop_as_black`.	—

`Policer_id` for APIs:
`bcm_policer_create`
`bcm_policer_set`
`bcm_policer_destroy`
	`Policer_id` is decoded by macro:
	`BCM_POLICER_ID_SET`
		The new macro decodes the `meter_idx` and database handle.

`bcm_policer_color_decision_set`	Removed. Similar API: `bcm_policer_primary_color_resolution_set`	—

Chapter 20: Statistic Interface

The statistic interface implementation in the BCM88690 is slightly different from the BCM88670.

Refer to the Statistic Interface chapter in the *Traffic Manager Programming Guide* for new SDK statistic interface design details.

The main differences are shown in the following table.

Headline	BCM88670	BCM88690
Network Interface	Dual-core mode/unified mode	Single mode, dual mode, or quad mode 400G/200G/100G
Object Interface	Two ingress/two egress, shared with CRPS	Dedicated objects for STIF: Four ingress, four egress per core
Flexible record format	No	Yes
Record size	64 or 60 bits	96, 80, 72, or 64
Flow control	No	Yes

New features are shown in the following table.

Headline	Description
Flexible record format	Configurable format of the record and its size.
Count ingress meters decisions	Count not only the final DP, but also the decision of each one of the ingress meters.
Metadata and TM attributes	Control which attributes will be added to the record, on top of the packet size and external object ID.
Flow control	Flow control toward the source (CGM/ETPP) to stop the records. May affect the total rate of the device.

New APIs are shown in the following table.

API	Description
`bcm_stat_stif_source_mapping_set` `bcm_stat_stif_source_mapping_get`	Configure the source mapping of the statistic interface sources and the logical ports.

`bcm_stat_control_set`
`bcm_stat_control_get`
`bcmStatControlStifFcEnable`
	Enable flow control towards the source to stop records.

`bcm_stat_pkt_size_adjust_select_set` `bcm_stat_pkt_size_adjust_select_get`	Select compensation packet size adjustment to determine whether to use header truncate.
`bcm_stat_counter_filter_group_set` `bcm_stat_counter_filter_group_get`	Activate filter drop reason groups.
`bcm_stat_stif_record_format_set` `bcm_stat_stif_record_format_get`	Configure the record format.

Changes to SOC properties from the BCM88670 to the BCM88690 include the following:

- Unsupported properties in the BCM88690:
  - `stat_if_enable_port`
  - `stat_if_tc_source`
  - `stat_if_core_mode`
  - `stat_if_rate; stat_if_scrubber_uc_dram_buffer_th_<thresh_id>`
  - `stat_if_scrubber_buffer_descr_th_<thresh_id>`
  - `stat_if_etpp_counter_mode`
  - `counter_engine_source`
- `stat_if_reports_per_packet` is renamed to `stat_if_pkt_size`
- `stat_if_billing_ingress_queue_stamp_enable` is made by using the record format API
- `stat_if_billing_ingress_drop_reason_enable` is made by using the record format API
- `stat_if_billing_filter_reports_ingress` is made by using the source mapping API
- `stat_if_billing_filter_reports_egress` is made by using the source mapping API

Chapter 21: Interrupts

Corrective actions were adjusted for the BCM88690.

There are no changes to the API.

Chapter 22: High Availability Warmboot

A new SOC property is required to inform the SDK that a unit supports warmboot.

New SOC Property Required to Enable Warmboot Support

```
warmboot_support=<on/off>
```

Autosync warmboot is not supported.

Unsupported Control for the bcm_switch_control_set() API

```
bcmSwitchControlAutoSync
```

S-cache CRC protection is not supported.

Unsupported Compilation Flag

```
SCACHE_CRC_CHECK
```

22.1 Crash Recovery

Crash recovery is no longer supported.

Unsupported Compilation Flag

```
CRASH_RECOVERY_SUPPORT
```

Unsupported Control for the bcm_switch_control_set() API

- `bcmSwitchControlAutoSync`
- `bcmSwitchCrashRecoveryMode`
- `bcmSwitchCrCommit`
- `bcmSwitchCrLastTransactionStatus`
- `bcmSwitchCrCouldNotRecover`

Unsupported SOC Properties

- `ha_hw_journal_mode`
- `ha_hw_journal_size`
- `ha_sw_journal_size`

Chapter 23: Stackable Systems

This feature is not supported for the BCM88690 family of devices.

BROADCOM®