# OMCI Performance Monitoring Drivers

# Revision History

| *Revision* | *Date* | *Change Description* |
|---|---|---|
| CPE-AN2500-R | 05/05/16 | Initial release |

# Table of Contents

# List of Figures

# List of Tables

# About This Document

## Purpose and Audience

The purpose of this document is to describe the interface between drivers API and OMCI (ONT Management and Control Interface) PM (Performance Monitoring) Application for the Broadcom GPON modem (BCM68XX) software configuration. This document is for software engineers designing applications with the BCM68XX family of devices.

## Acronyms and Abbreviations

In most cases, acronyms and abbreviations are defined on first use.

For a comprehensive list of acronyms and other terms used in Broadcom documents, go to: http://www.broadcom.com/press/glossary.php.

## Document Conventions

The following conventions may be used in this document:

| *Convention* | *Description* |
|---|---|
| **Bold** | User input and actions: for example, type **exit**, click **OK**, press **Alt+C** |
| `Monospace` | Code: `#include <iostream>`<br>HTML: `<td rowspan = 3>`<br>Command line commands and parameters: `wl [-l] <command>` |
| < > | Placeholders for *required* elements: enter your <username> or `wl <command>` |
| [ ] | Indicates *optional* command-line parameters: `wl [-l]`<br>Indicates bit and byte ranges (inclusive): [0:3] or [7:0] |

## References

The references in this section may be used in conjunction with this document.

**Note:** Broadcom provides customer access to technical documentation and software through its Customer Support Portal (CSP) and Downloads and Support site (see Technical Support).

For Broadcom documents, replace the “xx” in the document number with the largest number available in the repository to ensure that you have the most current version of the document.

| *Document (or Item) Name* | *Number* | *Source* |
|---|---|---|
| ***Broadcom Items*** | | |
| *[1] OMCI Message Capture and Playback Application Note* | CPE-AN27xx-R | CSP |

# Technical Support

Broadcom provides customer access to a wide range of information, including technical documentation, schematic diagrams, product bill of materials, PCB layout information, and software updates through its customer support portal (https://support.broadcom.com). For a CSP account, contact your Sales or Engineering support representative.

In addition, Broadcom provides other product support through its Downloads and Support site (http://www.broadcom.com/support/).

# Introduction

## Overview

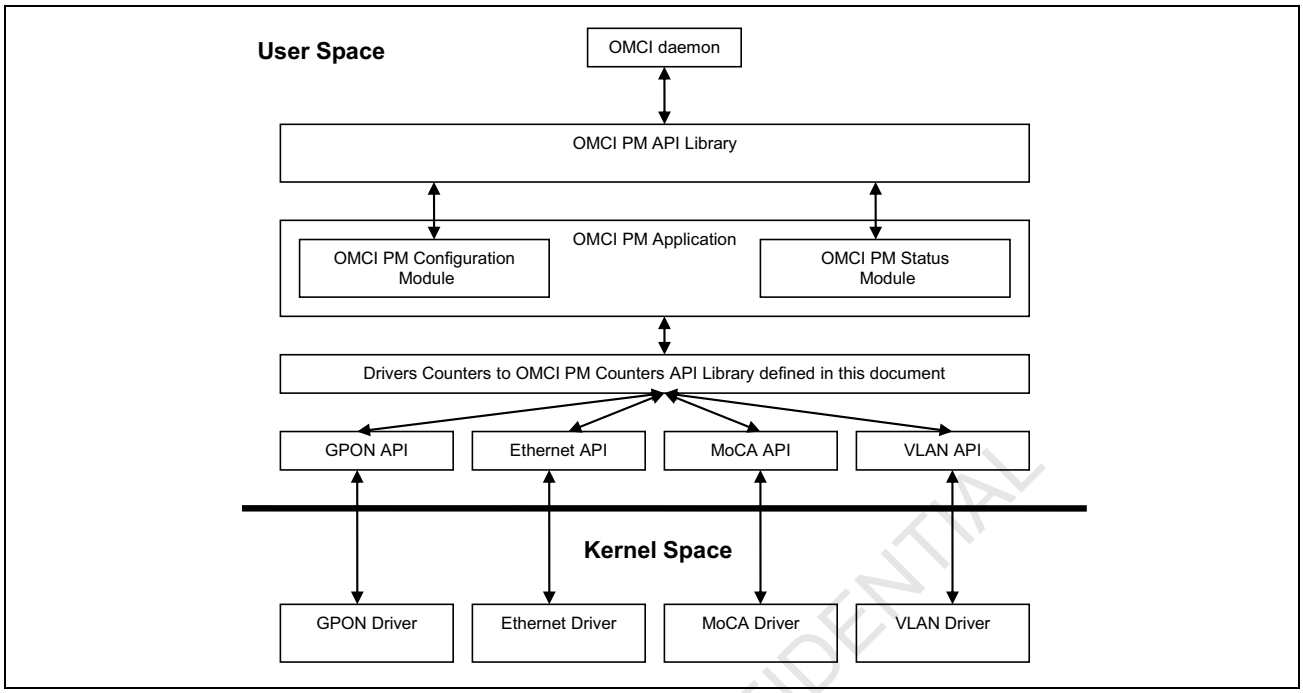
A fundamental goal for the development of the OMCI PM API was to provide a simple and efficient method to extract counters information from all the drivers API. The Drivers To OMCI PM API allows application developers to develop proprietary applications for retrieving counters in the GPON modem. Apart from vendor application developers, the Drivers To OMCI PM API is used by ONT Management and Control Interface Performance Monitoring task.

The Drivers To OMCI PM API allow software applications running on the BCM68XX GPON platform to programmatically retrieve the performance monitoring counters using simple programming interfaces. The following features are supported:

- Exports counters information to the applications as objects. Each object is uniquely identified and its relation to other objects is specified unambiguously.
- Provides seamless access across all applications that need to access counters information from all the drivers.
- Hides the implementation details and provides flexibility to the applications to retrieve the counters information from all the drivers.
- It also allows the separation of the Operating System (OS) dependent and OS independent sections of the code to make the future porting effort of the Drivers To OMCI PM API library much simpler, without any modifications to the applications that use the API.

A high-level view of the implementation and usage of the Drivers To OMCI PM API and the applications using the API is shown in Figure 1 on page 10. More details of the transactions are specified in later sections of this document. A sample configuration sequence is also specified in the later sections of the document.

**Figure 1: OMCI PM Architecture**




## Distribution

The Drivers To OMCI PM API consist of a set of C language functions. These functions are distributed in object form as a library or in source code form. C language header files are also distributed. They must be included in application source files that call Drivers To OMCI PM API functions. The header files contain constant definitions, data structures, and function prototypes that are described in this document.

## Portability

The Drivers To OMCI PM API are supported as an application on Linux. Since its interaction with the operating system is not extensive, all operating system dependent functionality is clearly isolated to make porting to other operating systems easy.

## Terminology

The following terms are used in this document.

- Object: An object is a logical or physical entity which consists of a collection of attributes. Objects in Drivers To OMCI PM API library can be read-only. Since these objects contain only counters information, the attributes of these objects are read-only and cannot be modified. An object defines the columns of the table.
- Object ID: An Object identifier is unique and is defined in Table 1 on page 11. The object ID of an object denotes a particular class of performance monitoring managed entity.
- Physical port ID: The Physical port identifier is unique. It is used to identify which physical port of the driver that counters information should be retrieved from.

# Drivers To OMCI PM API Summary

## Drivers To OMCI PM Objects

Table 1 is a list of all the currently available counter objects that are in the system.

***Table 1: OMCI PM Counter Objects***

| *Object ID* | *Object Structure* |
|---|---|
| ***GPON Performance Monitoring Group*** | |
| BCM_OMCI_PM_CLASS_GEM_PORT | BCM_OMCI_PM_GEM_PORT_COUNTER |
| BCM_OMCI_PM_CLASS_FEC | BCM_OMCI_PM_FEC_COUNTER |
| ***Ethernet Performance Monitoring Group*** | |
| BCM_OMCI_PM_CLASS_ENET | BCM_OMCI_PM_ETHERNET_COUNTER |
| BCM_OMCI_PM_CLASS_ENET2 | BCM_OMCI_PM_ETHERNET_2_COUNTER |
| BCM_OMCI_PM_CLASS_ENET3 | BCM_OMCI_PM_ETHERNET_3_COUNTER |
| BCM_OMCI_PM_CLASS_ENETDN | BCM_OMCI_PM_ETHERNET_UPDN_COUNTER |
| BCM_OMCI_PM_CLASS_ENETUP | BCM_OMCI_PM_ETHERNET_UPDN_COUNTER |
| ***MoCA Performance Monitoring Group*** | |
| BCM_OMCI_PM_CLASS_MOCA_ENET | BCM_OMCI_PM_MOCA_ETHERNET_COUNTER |
| BCM_OMCI_PM_CLASS_MOCA_INTF | BCM_OMCI_PM_MOCA_INTERFACE_COUNTER |
| ***GAL Ethernet Performance Monitoring Group*** | |
| BCM_OMCI_PM_CLASS_GAL_ENET | BCM_OMCI_PM_GAL_ETHERNET_COUNTER |
| ***Bridge Performance Monitoring Group*** | |
| BCM_OMCI_PM_CLASS_BRIDGE | BCM_OMCI_PM_BRIDGE_COUNTER |
| BCM_OMCI_PM_CLASS_BRIDGE_PORT | BCM_OMCI_PM_BRIDGE_PORT_COUNTER |
| ***Miscellaneous*** | |
| BCM_OMCI_PM_CLASS_RTP | BCM_OMCI_PM_RTP_COUNTER |
| BCM_OMCI_PM_CLASS_IPHOST | BCM_OMCI_PM_IP_HOST_COUNTER |



## GPON Performance Monitoring Group

The GPON performance monitoring group is a collection of all the objects that support the retrieving operation for the GPON physical interface. The following is a brief description of each object and its usage:

Object ID: BCM_OMCI_PM_CLASS_GEM_PORT

Description: This object collects performance monitoring data associated with a GEM port network CTP. Instances of this object can be retrieved by the Drivers To OMCI PM API. An instance of this object is associated with an instance of the physical path termination point GEM Port UNI.

Object ID: BCM_OMCI_PM_CLASS_FEC

Description: This object collects performance monitoring data associated with Forward Error Correction counters for GPON physical interface. Instances of this object can be retrieved by the OMCI PM API.

## Ethernet Performance Monitoring Group

The Ethernet performance monitoring group is a collection of all the objects that support configuration and status retrieving operations for the Ethernet physical interface. The following is a brief description of each object and its usage:

Object ID: BCM_OMCI_PM_CLASS_ENET

Description: This object collects some of the performance monitoring data for an Ethernet interface, as defined in the OMCI Ethernet performance monitoring history data ME. Instances of this object can be retrieved by the Drivers To OMCI PM API. An instance of this object is associated with an instance of the physical path termination point Ethernet UNI.

Object ID: BCM_OMCI_PM_CLASS_ENET2

Description: This object collects additional performance monitoring data for an Ethernet interface, as defined in the OMCI Ethernet performance monitoring history data 2 ME. Instances of this object can be retrieved by the Drivers To OMCI PM API. An instance of this object is associated with an instance of the physical path termination point Ethernet UNI.

Object ID: BCM_OMCI_PM_CLASS_ENET3

Description: This object collects performance monitoring data associated with an Ethernet interface, as defined in the OMCI Ethernet performance monitoring history data 3 ME. It includes parameters defined in the Ethernet statistics group of RFC 2819 that are not already covered by previously defined Ethernet monitoring MEs. The received direction is from the CPE toward the network (upstream). Instances of this object can be retrieved by the OMCI PM API. An instance of this object is associated with an instance of the physical path termination point Ethernet UNI.

Object ID: BCM_OMCI_PM_CLASS_ENETDN & BCM_OMCI_PM_CLASS_ENETUP

Description: This object collects performance monitoring data associated with an Ethernet interface, as defined in the OMCI Ethernet frame performance monitoring history data downstream and upstream MEs. Both are based on the Etherstats group of [IETF RFC 2819]. Instances of this object can be retrieved by the OMCI PM API. An instance of this object is associated with an instance of the physical path termination point Ethernet UNI.

## MoCA Performance Monitoring Group

The MoCA performance monitoring group is a collection of all the objects that support configuration and status retrieving operations for MoCA physical interface. The following is a brief description of each object and its usage:

Object ID: BCM_OMCI_PM_CLASS_MOCA_ENET

Description: This object collects the performance monitoring data for an MoCA Ethernet interface, as defined in the OMCI MoCA Ethernet performance monitoring history data ME. Instances of this object can be retrieved by the Drivers To OMCI PM API. An instance of this object is associated with an instance of the physical path termination point MoCA UNI.

Object ID: BCM_OMCI_PM_CLASS_MOCA_INTF

Description: This object collects the performance monitoring data for an MoCA interface, as defined in the OMCI MoCA interface performance monitoring history data ME. Instances of this object can be retrieved by the Drivers To OMCI PM API. This object has only current values, which are retrievable by get and get next operations; no history is retained. An instance of this object is associated with an instance of the physical path termination point MoCA UNI.

## GAL Ethernet Performance Monitoring Group

The GAL Ethernet group has a single object which is used to configure the GAL Ethernet GEM interworking termination point when the GEM layer provides Ethernet service. The following is a brief description and its usage:

Object ID: BCM_OMCI_PM_CLASS_GAL_ENET

Description: This object collects performance monitoring data associated with a GEM interworking termination point when the GEM layer provides Ethernet service, as defined in the OMCI GAL Ethernet performance monitoring history data ME. Instances of this object can be retrieved by the Drivers To OMCI PM API. An instance of this object is associated with an instance of the GEM interworking termination point.

## Bridge Performance Monitoring Group

The Bridge performance monitoring group is a collection of all the objects that support configuration and status retrieving operations for the MAC bridge. The following is a brief description of each object and its usage:

Object ID: BCM_OMCI_PM_CLASS_BRIDGE

Description: This object collects the performance monitoring data associated with the MAC bridge, as defined in the OMCI MAC bridge performance monitoring history data ME. Instances of this object can be retrieved by the Drivers To OMCI PM API. An instance of this object is associated with one instance of a MAC bridge service profile.

Object ID: BCM_OMCI_PM_CLASS_BRIDGE_PORT

Description: This object collects the performance monitoring data associated with the MAC bridge port, as defined in the OMCI MAC bridge port performance monitoring history data ME. Instances of this object can be retrieved by the Drivers To OMCI PM API. An instance of this object is associated with an instance of the a MAC bridge port.

## Miscellaneous

The following is a brief description of each object and its usage:

Object ID: BCM_OMCI_PM_CLASS_RTP

Description: This object collects the performance monitoring data associated with an RTP session, as defined in the OMCI RTP performance monitoring history data ME. Instances of this object can be retrieved by the Drivers To OMCI PM API. An instance of this object is associated with one instance of a PPTP POTS UNI.

Object ID: BCM_OMCI_PM_CLASS_IPHOST

Description: This object collects the performance monitoring data associated with an IP host, as defined in the OMCI IP host performance monitoring history data ME. Instances of this object can be retrieved by the Drivers To OMCI PM API. An instance of this object is associated with an instance of the an IP host config data.

# Drivers To OMCI PM API

This section describes the Drivers To OMCI PM API and its usage. These API are used by applications such as OMCI PM to get and get next counter objects.

## Function: bcm_omcipm_getCounters

This API call is used by the applications to retrieve the PM counters information of any object from the OMCI PM. The application user of this API will pass the object ID, a physical port ID to identify which port that counters information should be retrieved, and a pointer to a void type that can hold the counters structure. The applications must allocate the appropriate counters structure corresponding with the given object ID, and free this structure if necessary after using it.

The API will return the entry if found with success status, otherwise error status will be returned.

## Function: bcm_omcipm_getCountersNext

This API call is a special function that can only be used with OMCI PM object that supports get next operation such as BCM_OMCI_PM_CLASS_MOCA_INTF. It is used by the applications to retrieve the next PM counters information. The application user of this API will pass the object ID, a pointer to a physical port ID, and a pointer to a void type that can hold the counters structure.

This API will search the object table starting from the physical port ID plus one, and return the next available counters information. This API also returns the corresponding physical port ID if success. Otherwise it returns error status, and values in physical port ID and counters structure are undefined. The applications must allocate the appropriate counters structure corresponding with the given object ID, and free this structure if necessary after using it.

# Data Structures

## Class Structure

This section defines the enum structure that is used to identify OMCI PM class objects.

```
typedef enum {
   BCM_OMCI_PM_CLASS_GEM_PORT=0,
   BCM_OMCI_PM_CLASS_FEC,
   BCM_OMCI_PM_CLASS_ENET,
   BCM_OMCI_PM_CLASS_ENET2,
   BCM_OMCI_PM_CLASS_ENET3,
   BCM_OMCI_PM_CLASS_ENETDN,
   BCM_OMCI_PM_CLASS_ENETUP,
   BCM_OMCI_PM_CLASS_MOCA_ENET,
   BCM_OMCI_PM_CLASS_MOCA_INTF,
   BCM_OMCI_PM_CLASS_GAL_ENET,
   BCM_OMCI_PM_CLASS_BRIDGE,
   BCM_OMCI_PM_CLASS_BRIDGE_PORT,
   BCM_OMCI_PM_CLASS_RTP,
   BCM_OMCI_PM_CLASS_IPHOST,
   BCM_OMCI_PM_CLASS_MAX
} BCM_OMCI_PM_CLASS_ID;
```

# Counter Structures

This section defines the data structures that are used to retrieve the counters information of the OMCI PM objects. The only operation permitted on the counters objects is read.

The following standard type definitions are assumed:

- UINT8 – unsigned 8 bit integer or unsigned char
- UINT16 – unsigned 16 bit integer or unsigned short
- UINT32 – unsigned 32 bit integer or unsigned int

## BCM_OMCI_PM_GEM_PORT_COUNTER

```
typedef struct {
   OUT UINT32 transmittedGEMFrames;
   OUT UINT32 receivedGEMFrames;
   OUT UINT32 receivedPayloadBytes;
   OUT UINT32 transmittedPayloadBytes;
} BCM_OMCI_PM_GEM_PORT_COUNTER, *PBCM_OMCI_PM_GEM_PORT_COUNTER;
```

## BCM_OMCI_PM_FEC_COUNTER

```
typedef struct {
   OUT UINT32 correctedBytes;
   OUT UINT32 correctedCodeWords;
   OUT UINT32 uncorrectedCodeWords;
   OUT UINT32 totalCodeWords;
   OUT UINT32 fecSeconds;
} BCM_OMCI_PM_FEC_COUNTER, *PBCM_OMCI_PM_FEC_COUNTER;
```

## BCM_OMCI_PM_ETHERNET_COUNTER

```
typedef struct {
   OUT UINT32 fcsErrors;
   OUT UINT32 excessiveCollisionCounter;
   OUT UINT32 lateCollisionCounter;
   OUT UINT32 frameTooLongs;
   OUT UINT32 bufferOverflowsOnReceive;
   OUT UINT32 bufferOverflowsOnTransmit;
   OUT UINT32 singleCollisionFrameCounter;
   OUT UINT32 multipleCollisionsFrameCounter;
   OUT UINT32 sqeCounter;
   OUT UINT32 deferredTransmissionCounter;
   OUT UINT32 internalMacTransmitErrorCounter;
   OUT UINT32 carrierSenseErrorCounter;
   OUT UINT32 alignmentErrorCounter;
   OUT UINT32 internalMacReceiveErrorCounter;
} BCM_OMCI_PM_ETHERNET_COUNTER, *BCM_OMCI_PM_ETHERNET_COUNTER;
```

## BCM_OMCI_PM_ETHERNET_2_COUNTER

```
typedef struct {
   OUT UINT32 pppoeFilterFrameCounter;
} BCM_OMCI_PM_ETHERNET_2_COUNTER, *PBCM_OMCI_PM_ETHERNET_2_COUNTER;
```

## BCM_OMCI_PM_ETHERNET_3_COUNTER

```
typedef struct {
   OUT UINT32 dropEvents;
   OUT UINT32 octets;
   OUT UINT32 packets;
   OUT UINT32 broadcastPackets;
   OUT UINT32 multicastPackets;
   OUT UINT32 undersizePackets;
   OUT UINT32 fragments;
   OUT UINT32 jabbers;
   OUT UINT32 packets64Octets;
   OUT UINT32 packets127Octets;
   OUT UINT32 packets255Octets;
   OUT UINT32 packets511Octets;
   OUT UINT32 packets1023Octets;
   OUT UINT32 packets1518Octets;
} BCM_OMCI_PM_ETHERNET_3_COUNTER, *PBCM_OMCI_PM_ETHERNET_3_COUNTER;
```

## BCM_OMCI_PM_CLASS_ENETDN and BCM_OMCI_PM_CLASS_ENETUP

```
typedef struct {
   OUT UINT32 dropEvents;
   OUT UINT32 octets;
   OUT UINT32 packets;
   OUT UINT32 broadcastPackets;
   OUT UINT32 multicastPackets;
   OUT UINT32 crcErroredPackets;
   OUT UINT32 undersizePackets;
   OUT UINT32 oversizePackets;
   OUT UINT32 packets64Octets;
   OUT UINT32 packets127Octets;
   OUT UINT32 packets255Octets;
   OUT UINT32 packets511Octets;
   OUT UINT32 packets1023Octets;
   OUT UINT32 packets1518Octets;
} BCM_OMCI_PM_ETHERNET_UPDN_COUNTER, *PBCM_OMCI_PM_ETHERNET_UPDN_COUNTER;
```

## BCM_OMCI_PM_MOCA_ETHERNET_COUNTER

```
typedef struct {
   OUT UINT32 incomingUnicastPackets;
   OUT UINT32 incomingDiscardedPackets;
   OUT UINT32 incomingErroredPackets;
   OUT UINT32 incomingUnknownPackets;
   OUT UINT32 incomingMulticastPackets;
   OUT UINT32 incomingBroadcastPackets;
   OUT UINT32 incomingOctets;
   OUT UINT32 outgoingUnicastPackets;
   OUT UINT32 outgoingDiscardedPackets;
   OUT UINT32 outgoingErroredPackets;
   OUT UINT32 outgoingUnknownPackets;
   OUT UINT32 outgoingMulticastPackets;
   OUT UINT32 outgoingBroadcastPackets;
   OUT UINT32 outgoingOctets_hi;
   OUT UINT32 outgoingOctets_low;
} BCM_OMCI_PM_MOCA_ETHERNET_COUNTER, *PBCM_OMCI_PM_MOCA_ETHERNET_COUNTER;
```

## BCM_OMCI_PM_MOCA_INTERFACE_COUNTER

```
typedef struct {
   OUT UINT32 phyTxBroadcastRate;
   OUT UINT32 phyTxRate;
   OUT UINT32 txPowerControlReduction;
   OUT UINT32 phyRxRate;
   OUT UINT32 rxPowerLevel;
   OUT UINT32 phyRxBroadcastRate;
   OUT UINT32 rxBroadcastPowerLevel;
   OUT UINT32 txPackets;
   OUT UINT32 rxPackets;
   OUT UINT32 erroredMissedRxPackets;
   OUT UINT32 erroredRxPackets;     OUT UINT8 mac[BCM_OMCI_MAC_STR_SIZE];
} BCM_OMCI_PM_MOCA_INTERFACE_COUNTER, *PBCM_OMCI_PM_MOCA_INTERFACE_COUNTER;
```

## BCM_OMCI_PM_GAL_ETHERNET_COUNTER

```
typedef struct {
   OUT UINT32 discardedFrames;
   OUT UINT32 transmittedFrames;
   OUT UINT32 receivedFrames;
} BCM_OMCI_PM_GAL_ETHERNET_COUNTER, *PBCM_OMCI_PM_GAL_ETHERNET_COUNTER;
```

## BCM_OMCI_PM_MAC_BRIDGE_COUNTER

```
typedef struct {
    OUT   UINT32 learningDiscaredEntries;
} BCM_OMCI_PM_MAC_BRIDGE_COUNTER, *PBCM_OMCI_PM_MAC_BRIDGE_COUNTER;
```

## BCM_OMCI_PM_MAC_BRIDGE_PORT_COUNTER

```
typedef struct {
   OUT   UINT32 forwardedFrames;
   OUT   UINT32 delayDiscardedFrames;
   OUT   UINT32 mtuDiscardedFrames;
   OUT   UINT32 receivedFrames;
   OUT   UINT32 receivedDiscardedFrames;
} BCM_OMCI_PM_MAC_BRIDGE_PORT_COUNTER, *PBCM_OMCI_PM_MAC_BRIDGE_PORT_COUNTER;
```

## BCM_OMCI_PM_RTP_COUNTER

```
typedef struct {
   OUT UINT32 rtpErrors;
   OUT UINT32 packetLoss;
   OUT UINT32 maxJitter;
   OUT UINT32 maxTimeBetweenRtcpPackets;
   OUT UINT32 bufferUnderflows;
   OUT UINT32 bufferOverflows;
} BCM_OMCI_PM_RTP_COUNTER, *PBCM_OMCI_PM_RTP_COUNTER;
```

## BCM_OMCI_PM_IP_HOST_COUNTER

```
typedef struct {
   OUT UINT32 icmpErrors;
   OUT UINT32 dnsErrors;
} BCM_OMCI_PM_IP_HOST_COUNTER, *PBCM_OMCI_PM_IP_HOST_COUNTER;
```

# Functions

The Drivers to OMCI PM API are used between an OMCI PM task and the various drivers API such as the GPON API, Ethernet API, MoCA API, etc.

## bcm_omcipm_getCounters

**Syntax:**

```
BCM_OMCI_PM_STATUS bcm_omcipm_getCounters(BCM_OMCI_PM_CLASS_ID      classId,
                                          UINT16               physPortId,
                                          void                 *counters);
```

**Parameters:**

*BCM_OMCI_PM_CLASS_ID*

[in] classId. Constant that represents an object. For values see Table 1 on page 11.

*physPortId*

[in] UINT16. Physical port identifier of the driver.

**counters*

[out] void. Pointer to the object that contains counters information. It is casted to a void type.

## bcm_omcipm_getCountersNext

**Syntax:**

```
BCM_OMCI_PM_STATUS bcm_omcipm_getCountersNext(BCM_OMCI_PM_CLASS_ID classId,
                                          UINT16               physPortId,
                                          void                 *counters);
```

**Parameters:**

*BCM_OMCI_PM_CLASS_ID*

[in] classId. Constant that represents an object. For values see Table 1 on page 11.

*physPortId*

[in] UINT16. Physical port identifier of the driver.

**counters*

[out] void. Pointer to the object that contains counters information. It is casted to a void type.

Web: www.broadcom.com
Corporate Headquarters: San Jose, CA


CPE-AN2500-R May 5, 2016