

RDPA Traffic Management Operation Guide

Jpei Confidential

Revision History

Revision	Date	Change Description
RDPA-AN100-R	10/30/13	Initial release



Broadcom Corporation 5300 California Avenue Irvine, CA 92617

© 2013 by Broadcom Corporation All rights reserved Printed in the U.S.A.

Broadcom®, the pulse logo, Connecting everything®, and the Connecting everything logo are among the trademarks of Broadcom Corporation and/or its affiliates in the United States, certain other countries and/ or the EU. Any other trademarks or trade names mentioned are the property of their respective owners.

Table of Contents

About This Document		5
Purpose and Audience		5
Acronyms and Abbreviations		5
Document Conventions		5
Technical Support		5
RDPA TM Architecture		6
Code Examples		7
RDPA Control API		7
RDPA Command Driver		8
RDPA TM Middleware		8
RDPA Traffic Models	×	9
Strict Priority		9
Weighted Round Robin		11
QoS Initialization and Configuration		11
Configuration Modes		11
RDPA Control APIs		16
\$\langle \langle \lang		

RDPA Application Note List of Figures

List of Figures

Figure 1:	RDPA TM Architecture	6
Figure 2:	Strict Priority Scheduler	
Figure 3:	Weighted Round Robin Scheduler	11
Figure 4:	QoS Initialization and Provisioning	12
Figure 5:	QoS Queue Management Configuration	12
Figure 6:	QoS Queue Configuration	13
Figure 7:	QoS Setup Screen	14
Figure 8:	QoS Queue Configuration	15
	QoS Queue Configuration	

RDPA Application Note About This Document

About This Document

Purpose and Audience

This document explains the Runner Data Path API (RDPA) Traffic Management (TM) architecture and QoS queue configuration via the CMS Web UI. It is aimed for software engineers using RDPA in their system.

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> HTML: Command line commands and parameters: wl [-1] <command/></iostream>		
<>	Placeholders for required elements: enter your <username> or wl <command/></username>		
[]	Indicates optional command-line parameters: w1 [-1] Indicates bit and byte ranges (inclusive): [0:3] or [7:0]		

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/).

RDPA Application Note RDPA TM Architecture

RDPA TM Architecture

Figure 1 shows the RDPA TM architecture.

CMS QoS Management
CMS QoS Management
User Space Utility QoS (TBD)

RDPA Control API

User Space

User Space

RDPA Cm Driver

RDPA Tm Middleware

RDPA

RDPA

Firmware

Figure 1: RDPA TM Architecture

CMS QoS Management represents an interface between TR-69 management (or Web UI) and the CMS Core APIs. This component is built around the configuration transactions, is stateless and is platform independent.

The RDPA QoS Interface represents a platform dependent glue layer between CMS and RDPA TM. This component translates the CMS transactions to the specific QoS model through the RDPA Control API.

RDPA Control API is the user space API which is responsible for the parameters encapsulation and the IOCTL calls. There is no decision making logic in this component. See the code examples below.

The RDPA Command Driver is a Linux Kernel character device which is responsible for the commands dispatching from/to the user space. The user space IOCTL call is received here and is redirected to the appropriate handler routine.

RDPA TM Middleware is the RDPA/BDMF abstraction layer. The actual RDPA/BDMF calls are encapsulated here. This component is responsible for the RDPA objects allocation/deletion, for the RDPA calls order and internal logic, etc.

RDPA Application Note Code Examples

The RDPA TM User Space Utility is the user space utility that encapsulates the RDPA TM IOCTL calls in case of a non CMS third-party system. We assume that the third-party residential gateway software will use the tc Linux utility to configure QoS models in Linux and Flow Cache and the rdpactl utility to configure the Runner accelerator QoS. There is also an option to avoid using rdpactl by directly calling the RDPA TM IOCTLs. This creates an open architecture which does not depend on specific customer implementation.

Code Examples

RDPA Control API

```
int rdpaCtl TmConfig(
   int rootTmId,
   int portId,
   int dir,
   int level,
   int arbiterMode,
   int priority,
   int weight,
   int *pTmId)
   int rc = 0; The
   rdpa_drv_ioctl_tm_t tm;
   memset(&tm, 0, sizeof(rdpa_drv_ioctl_tm_t));
   tm.cmd = RDPA_IOCTL_TM_CMD_TM_CONFIG;
   tm.root_tm_id = rootTmId;
   tm.port_id = portId;
   tm.dir = dir;
   tm.level = level;
   tm.arbiter_mode = arbiterMode
   tm.priority = priority;
   tm.weight = weight;
   rc = sendTmCommand(&tm)
   if (!rc) {
       *pTmId = tm.tm id;
       rdpaCtl_debug("NEW TM ID %d, dir %d, level %d, arbiterMode %d",
           *pTmId, dir, level, arbiterMode);
   return rc;
}
```

RDPA Application Note Code Examples

RDPA Command Driver

```
static long rdpa_cmdIoctl(struct file *filep, unsigned int command, unsigned long arg)
  rdpaDrvIoctl_t cmd;
  int ret = RDPA_DRV_SUCCESS;
   switch( cmd )
       case RDPA_IOC_TM:
       {
           ret = rdpa_cmd_tm_ioctl(arg);
           break;
       default:
                                              OKIO
   }
   return ret;
} /* rdpa_cmdIoctl */
RDPA TM Middleware
case RDPA_IOCTL_TM_CMD_TM_REMOVE: {
  bdmf_error_t rc = BDMF_ERR_OK;
  bdmf object handle sched = NULL;
   rdpa_egress_tm_key_t tm_key = {};
  tm_key.dir = tm.dir;
  tm_key.index = tm.tmId;
  if ((rc = rdpa_egress_tm_get(&tm_key, &sched))) {
      ret = RDPA_DRV_ERROR;
      goto ioctl_exit;
   if (sched) {
      bdmf_put(sched);
      bdmf_destroy(sched)
   }
}
```

RDPA Application Note RDPA Traffic Models

RDPA Traffic Models

There are various traffic management models supported by RDPA. Currently the following models are implemented by CMS TM:

- Strict Priority (SP)
- Weighted Round Robin (WRR)

Strict Priority

The SP scheduling scheme is created using a single Root TM and a several secondary level TM objects, as shown in Figure 2. The Root TM defines the scheduler mode, in this case SP. The TM objects are configured with Rate Limiter value.

Figure 2: Strict Priority Scheduler **Subsidiary TM Objects** Egress_tm_2 Queue 0 Root TM Object Level – Queue Mode - Disable Rate Limiter = xxx Egress_tm_1 Level - TM Owned by Mode - SP Egress_tm_3 Queue 0 subsidiary[] = Level - Queue Egress_tm_2, Mode – Disable Egress_tm_3, Rate Limiter = yyy Egress_tm_4 Egress_tm_4 Queue 0 Level - Queue Mode – Disable Rate Limiter = zzz

RDPA Application Note RDPA Traffic Models

Example: The code below is a command line example for two queues.

```
# bs /b/e egress_tm
Object: egress_tm/dir=us,index=39. Object type: egress_tm. Owned by: egress_tm/dir=us,index=37
_____
                dir : us
              index: 39
              level : queue
                                                                    Subsidiary TM Object
               mode : disable
         overall_rl : no
             enable : yes
                 rl : {af=0,be=0}
queue_cfg[0] : {queue_id=1,drop_threshold=127,weight=0,drop_alg=dt,red_threshold=0,
red_percent=0}
queue_stat[{channel={tcont/index=0},queue_id=1}] : 0
queue_stat[{channel={tcont/index=1},queue_id=1}] : 0
             weight: 0
Object: egress_tm/dir=us,index=38. Object type: egress_tm. Owned by: egress_tm/dir=us,index=37
_____
                dir : us
              index: 38
              level : queue
                                                                    Subsidiary TM Object
               mode : disable
         overall rl : no
             enable : yes
                 rl : {af=0,be=0}
queue_cfg[0] : {queue_id=0,drop_threshold=127,weight=0,drop_alg=dt,red_threshold=0,
red_percent=0}
queue stat[{channel={tcont/index=0}, queue id≠0}]
queue_stat[{channel={tcont/index=1}, queue_id=0}] ): 0
             weight: 0
Object: egress_tm/dir=us,index=37. Object type: egress_tm. Owned by: port/index=wan0
_____
                dir : us
              index: 37
              level : egress
                                                                       Root TM Object
               mode : sp
         overall_rl : no
             enable : yes
                 rl ; {af=0,be=0}
queue_stat[{channel={tcont/index=0},queue_id=1}] : 0
queue_stat[{channel={tcont/index=0},queue_id=0}] : 0
queue_stat[{channel={tcont/index=1},queue_id=1}] : 0
queue_stat[{channel={tcont/index=1},queue_id=0}] : 0
  *other elements not shown*
      subsidiary[6] : {egress_tm/dir=us,index=39}
       subsidiary[7] : {egress_tm/dir=us,index=38}
             weight: 0
```

Weighted Round Robin

The WRR scheduling scheme is built using a single Root TM and a several secondary level TM objects, similar to the SP scheme, see Figure 3. The Root TM defines a scheduler mode – WRR. The TM objects are configured with a queue weight. Weight is defined per TM object and queue weight is not used in this model.

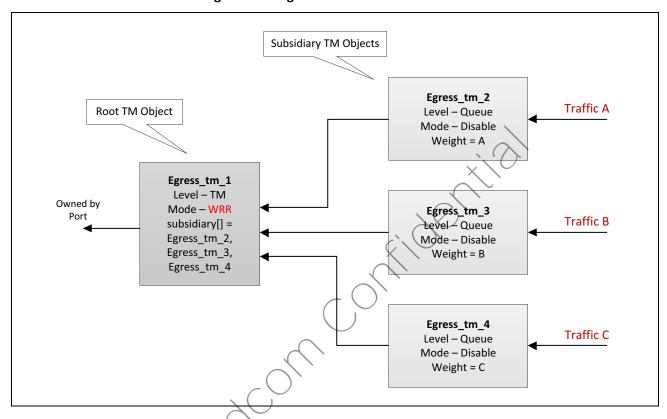


Figure 3: Weighted Round Robin Scheduler

QoS Initialization and Configuration

Configuration Modes

CMS TM and QoS configuration are encapsulated in the RDPA TM module. Default QoS configuration is performed during the module initialization at system startup. After startup the CMS System enters the Default QoS Mode.

During the CMS configuration session (TR-69 or Web UI) QoS parameters are provisioned. This is the QoS Provisioned Mode. QoS models for this mode are described in "RDPA Traffic Models".

If the QoS configuration is removed then the system returns to the default QoS mode. The default mode QoS model consists of a Strict Priority scheduler where a single TM object holds eight queues. Figure 4 shows the QoS initializing and provisioning flow.

Default QoS Mode

TR-69 QoS
Provisioning

QoS Provisioned Mode

Figure 4: QoS Initialization and Provisioning

Configuring A QoS Queue

This section gives an example of how to configure the QoS queues using the CMS Web UI.

Follow the procedure to define the QoS.

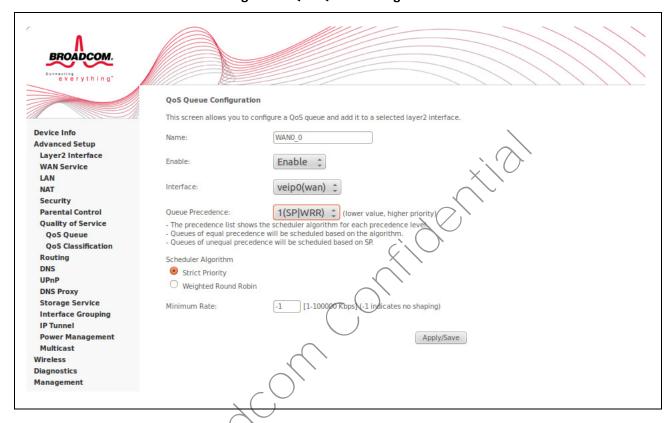
1. Open the Web UI, select the Quality of Service tab, Figure 5.

BROADCOM. everything" QoS -- Queue Management Configuration If Enable QoS checkbox is selected, choose a default DSCP mark to automatically mark incoming traffic without reference to a particular classifier. Click 'Apply/Save' **Device Info** Advanced Setup Layer2 Interface Note: If Enable Qos c neckbox is not selected, all QoS will be disabled for all interfaces. WAN Service Note: The default DSCP mark is used to mark all egress packets that do not match any classification rules. LAN NAT Security **Parental Control** Quality of Service QoS Queue Select Default DSCP Mark No Change(-1) **QoS Classification** Routing DNS Apply/Save UPnP **DNS Proxy** Storage Service Interface Grouping **IP Tunnel** Multicast Wireless Diagnostics Management

Figure 5: QoS Queue Management Configuration

- 2. Check the Enable QoS checkbox.
- 3. Click Apply/Save.
- **4.** Open the QoS Queue tab, click **Add Queue**. The QoS Queue Configuration screen opens Figure 6.

Figure 6: QoS Queue Configuration



- **5.** Type a queue name.
- **6.** Choose an interface:
 - veip0(wan) for GRON
 - EPON0 for EPON
- **7.** Select the queue priority. The lower value is a higher priority.
- 8. Select a scheduling algorithm.
 - Strict Priority
 - · Weighted Round Robin

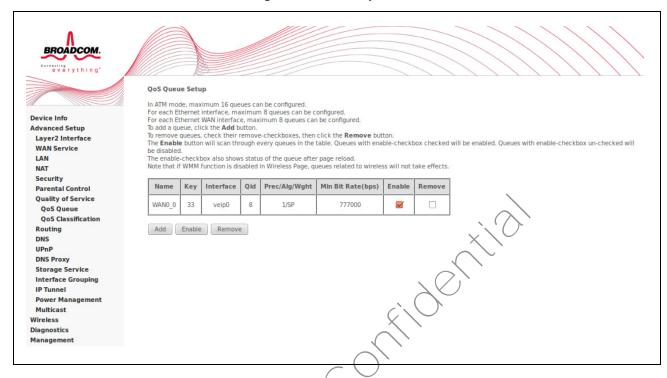


Note: The scheduling algorithm for subsequent queues is taken from this first configured queue. All other queues will either be SP or WRR.

9. For SP algorithm type in the Rate controller rate in kilobits per second or leave -1 to skip the rate. For WRR type in the queue weight.

10. Click Apply/Save. The QoS Setup screen is displayed Figure 7.

Figure 7: QoS Setup Screen



To add another queue follow the procedure from Step 11.

11. Click Add. The QoS Queue Configuration screen opens, but it displays the previously chosen configuration, as shown in Figure 8.

everything' **QoS Queue Configuration** This screen allows you to configure a QoS queue and add it to a selected layer2 interface. Device Info Name: WANO_5 Advanced Setup Layer2 Interface Enable: Enable WAN Service LAN Interface: veip0(wan) Quality of Service QoS Queue Queue Precedence: QoS Classification Routing - The precedence list shows the scheduler algorithm for sach precedence level. Queues of equal precedence will be scheduled based on the algorithm. DNS Queues of unequal precedence will be scheduled based on SP. UPnP **DNS Proxy** Minimum Rate: [1-100000 Kbps] (-1 indicates no shaping) Diagnostics Management

Figure 8: QoS Queue Configuration

- **12.** Type a queue name.
- **13.** Choose an interface:
 - veip0(wan) GPON
 - EPONO EPON
- 14. Select the queue priority. Previous queue priorities do not appear in the list.



Note: The algorithm is taken from the first configured queue: SP or WRR. All subsequent queues take the algorithm of the first queue.

- **15.** In this example, the SP algorithm was previously selected, so the rate controller rate is required. Enter kilobits per second or leave -1 to skip the rate.
- 16. Continue from Step 11 until all required QoS queues have been added and configured.

The downstream queue configuration is very similar, with the following exceptions:

- The SP only scheduling algorithm is available in the Downstream direction (no WRR).
- There is no rate per queue configuration.

RDPA Application Note RDPA Control APIs

RDPA Control APIs

```
* Function: rdpaCtl_ GetRootTmByIfname
  portId,
                 // IN: RDPA Port Id
  pRootTmId,
                  // IN: Root Traffic Management ID
                  // OUT: TM Found/Not Found
  pbFound
int rdpaCtl_GetRootTmByPortId(
  int portId,
  int *pRootTmId,
  BOOL *pbFound);
Function: rdpaCtl_GetTmByQid
            // IN: RDPA Port Id
  portId,
                 // IN: Management Queue Id
  qId,
                 // OUT: Traffic Management Object ID
  pTmId,
  pbFound,
                  // OUT: Tm ID Found or NOT
***************
int rdpaCtl_GetTmByQid(
  int portId,
  int qId,
  int *pTmId ,
  BOOL *pbFound);
/***********
* Function: rdpaCtl_RootTmConfig
                  // IN: RDPA Port Id
 portId
* dir,
                  // IN: Tm direction set
* level,
                  // IN: Tm type
                  // IN: Tm arbiter mode (SP, WRR, etc.)
* arbiterMode
* pRootTmId,
                  // OUT: Root Traffic Management ID
int rdpaCtl_RootTmConfig(
  int
       portId,
  int
        dir,
  int
        level,
  int
       arbiterMode
  int
       * pRootTmId)
```

RDPA Application Note RDPA Control APIs

```
* Function: rdpaCtl_TmConfig
* rootTmId,
                       // IN: Root Traffic Management (Tm) ID
 portId
                       // IN: RDPA interface (port)
* dir,
                      // IN: Tm direction set
* level,
                      // IN: Tm type ()
* arbiterMode
                      // IN: Tm arbiter mode (SP, WRR, etc.)
* priority,
                      // IN: Tm Priority (for SP)
* weight,
                       // IN: TM Weight (for WRR)
* pTmId,
                       // OUT: Traffic Management Object ID
int rdpaCtl_Config(
          rootTmId,
   int
   int
          portId,
   int
          dir,
   int
          level,
   int
          arbiterMode,
   int
          priority,
   int
          weight,
   int
         *pTmId);
/**************
 Function: rdpaCtl_RootTmRemove
    tmId,
                           // IN
                                     : Traffic Management (Tm) ID
    dir,
                           // IN
                                    : Tm direction set
int rdpaCtl_RootTmRemove(
   int tmId,
   int dir);
/**********
* Function: rdpaCtl_TmRemove
                           // IN: Interface name
   ifName,
                           // IN: Traffic Management (Tm) ID
   tmId,
                           // IN: Tm direction set
int rdpaCtl_TmRemove(
   int tmId,
   int dir);
```

RDPA Application Note RDPA Control APIs

```
* Function: rdpaCtl_QueueConfig
* portId
                  // IN: RDPA interface (port)
                   // IN: Traffic Management (Tm) ID
* tmId,
                  // IN: Queue ID
* q_id
                  // IN: Queue size
* qsize,
* weight,
                  // IN: TM Weight (for WRR)
* dir,
                  // IN: Tm direction set
* shapingRate,
                  // IN: af_rate
* shapingBurstSize
                  // IN: be rate
* weight
                   // IN: Queue weight
int rdpaCtl_QueueConfig(
  int portId,
  int tmId,
  int q_id,
  int qsize,
  int weight,
  int dir,
  int shaping_rate,
  int shaping_burst_size);
/***************
 Function: rdpaCtl_QueueRemove
                    // IN
                             : RDPA interface (port)
    portId
                          : Traffic Management (Tm) ID
                     // IN
   tmId,
                             : Tm direction set
                     // IN
    dir,
                    // IN
                            : Queue ID/
* **************
int rdpaCtl_QueueRemove(
               3/020/01
  int portId,
  int tmId,
  int dir,
  int qid);
```



Broadcom® Corporation reserves the right to make changes without further notice to any products or data herein to improve reliability, function, or design.

Information furnished by Broadcom Corporation is believed to be accurate and reliable. However, Broadcom Corporation does not assume any liability arising out of the application or use of this information, nor the application or use of any product or circuit described herein, neither does it convey any license under its patent rights nor the rights of others.

everything®

BROADCOM CORPORATION

5300 California Avenue Irvine, CA 92617 © 2013 by BROADCOM CORPORATION. All rights reserved. Phone: 949-926-5000 Fax: 949-926-5203

E-mail: info@broadcom.com Web: www.broadcom.com