QCS-AX/QCS-AX2: Netconsole

UM70055/D

INTRODUCTION

Netconsole enables kernel log messages to be sent to one or several receiving clients over UDP. This allows debugging work to be conducted without the traditional need for a serial console or disk logging. Netconsole can be built as a Linux module or built−in to the kernel itself and enabled at run−time or boot−time. Starting from release R6.5, QCS provides Netconsole as a module by default. The below sections first describe basic APDUT and log server setup, followed by an advanced section for dynamic and multiple log server operations. Additional Netconsole documentation can be found on kernel.org.

ABBREVIATIONS AND ACRONYMS

Abbreviations and Acronyms	Description
APDUT	Access Point Device Under Test
MAC	Medium Access Control
QCS	Quantenna Connectivity Solutions Division
QCS−AX	All variants of the QCS−AX family of chipsets
QCS−AX2	All variants of the QCS−AX2 family of chipsets
UDP	User Data Protocol

Netconsole Definition

```
netconsole=[+][src−port]@[src−ip]/[<dev>],[tgt−port]@<tgt−ip>/[tgtmacaddr]
```

- + if present, enable extended console support
- src−port source for UDP packets (defaults to 6665)
- src−ip source IP to use (interface address)
- dev network interface (eth0)
- tgt−port port for logging agent (6666)
- tgt−ip IP address for logging agent
- tgt−macaddr ethernet MAC address for logging agent (broadcast)

APDUT SETUP

Run−time Enablement

Load the Netconsole module with the desired parameters by manually inputting Netconsole in command prompt or by scripting.

For example:

```
# modprobe netconsole
netconsole="@/br0,6666@192.168.1.100/"
```

This will bring up the Netconsole module on the APDUT transmitting the log messages out of br0 UDP port 6665 (default) to the log server at 192.168.1.100 that is listening on UDP port 6666. Multiple source/target pairs can be specified.

Boot−time Enablement

Netconsole can be enabled automatically during boot−up with common QCS enabled scripting methods.

- Create a text file named tweak_qcomm at /mnt/jffs2/.
- Add the desired parameters. For example, the above modprobe command can be copied exactly as shown.
- Make sure the newly created tweak_qcomm file is executable.

```
# chmod +x tweak_qcomm
```

- Reboot the system.

Prints like below indicate Netconsole is inserted:

```
load_module: netconsole at 0x8b016000, size 0x127a
netpoll: netconsole: local port 6665
netpoll: netconsole: local IPv4 address 0.0.0.0
netpoll: netconsole: interface 'br0'
netpoll: netconsole: remote port 6666
netpoll: netconsole: remote IPv4 address 192.168.1.100
netpoll: netconsole: remote ethernet address ff:ff:ff:ff:ff:ff
netpoll: netconsole: local IP 192.168.1.244
console [netcon0] enabled
netconsole: network logging started
```

The Netconsole module can be removed and reinserted as needed without requiring reboot.

```
# modprobe −r netconsole
```

To verify the module information:

```
# modinfo netconsole
```

To set the debug level for maximum output:

```
# dmesg −n7
```

Log Server

The log server that receives UDP packets from the APDUT has flexibility on what logging tool is used. Common implementations use syslogd, netcat, or socat. For

© Semiconductor Components Industries, LLC, 2021

example, on Ubuntu (16.04, 20.04) using netcat, from a terminal window run:

```
# nc −u −l −p 6666
```

Output that follows should match up exactly to the prints that can be seen from dmesg on the APDUT. A simple bring down and back up of a WiFi interface will trigger debug messages that can be verified on the log server.

For example, these interface down/up prints will show identically on both the APDUT logs and the remote server terminal window:

br0: port 2(wifi0_0) entered disabled state
br0: port 2(wifi0_0) entered blocking state
br0: port 2(wifi0_0) entered forwarding state

ADVANCED OPERATIONS

Multiple Log Servers

The Netconsole module can be brought up in a configuration that supports multiple log servers. The command syntax separating each log server destination is the semi−colon, ";".

```
# modprobe netconsole
netconsole="@/br0,6666@192.168.1.100/;@/br0,6667@192.
168.1.101/"
```

Dynamic Netconsole

Dynamic reconfigurability is a useful addition to Netconsole that enables remote logging targets to be dynamically added or removed, and have parameters reconfigured at run time from a configfs based userspace interface. Dynamic support must be added by means of changing the Linux kernel build parameters within the SDK and generating a new image:

- Edit "pearl_10gax_config" or "jade_config" file located at /linux−4.19.35/arch/arc/configs/
- Find: # CONFIG_CONFIGFS_FS is not set
  Replace with: CONFIG_CONFIGFS_FS=y
  CONFIG_NETCONSOLE_DYNAMIC=y
- CONFIG_CONFIGFS_FS adds a userspace driven configuration filesystem needed for Netconsole dynamic control. Three additional build options will be triggered when CONFIGFS_FS is first used in the build, all of which can be set to "N", no.
  - NVMe Target Support
  - OCFS2 File Support
  - Distributed Lock Manager DLM

Dynamic Configuration

With an image that has dynamic Netconsole support added and the module loaded, mount configfs and create the target directory.

```
# mount none −t configfs /sys/kernel/config
# mkdir /sys/kernel/config/netconsole/target
```

Once the target directory is created, Netconsole will automatically populate with the configuration parameters in the target directory.

```
# ls /sys/kernel/config/netconsole/target/
dev_name  extended  local_mac  remote_ip  remote_port
enabled   local_ip  local_port  remote_mac
```

By populating the configuration parameters, desired settings can be applied. For example:

```
cat enabled                    # check enabled/disabled
echo 0 > enabled               # disable the target
echo br0 > dev_name            # set local interface
echo 192.168.1.100 > remote_ip # set remote IP
echo 6666 > remote_port        # set remote port
echo 1 > enabled               # enable target again
```

Multiple targets can be created and used at the same time. Use the mkdir procedure previously shown and populate the parameters for each target.

```
# mkdir /sys/kernel/config/netconsole/target<N>
```

Targets that are created are not persistent across a reboot. They will need to be re−created after each reboot which can be done by a tweak_qcomm script.

QUANTENNA is a registered trademark of Semiconductor Components Industries, LLC (SCILLC) or its subsidiaries in the United States and/or other countries. All other brand names and product names appearing in this document are registered trademarks or trademarks of their respective holders.

onsemi, ONSEMI , and other names, marks, and brands are registered and/or common law trademarks of Semiconductor Components Industries, LLC dba "**onsemi**" or its affiliates and/or subsidiaries in the United States and/or other countries. **onsemi** owns the rights to a number of patents, trademarks, copyrights, trade secrets, and other intellectual property. A listing of **onsemi's** product/patent coverage may be accessed at www.onsemi.com/site/pdf/Patent−Marking.pdf. **onsemi** reserves the right to make changes at any time to any products or information herein, without notice. The information herein is provided "as−is" and **onsemi** makes no warranty, representation or guarantee regarding the accuracy of the information, product features, availability, functionality, or suitability of its products for any particular purpose, nor does **onsemi** assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation special, consequential or incidental damages. Buyer is responsible for its products and applications using **onsemi** products, including compliance with all laws, regulations and safety requirements or standards, regardless of any support or applications information provided by **onsemi**. "Typical" parameters which may be provided in **onsemi** data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. **onsemi** does not convey any license under any of its intellectual property rights nor the rights of others. **onsemi** products are not designed, intended, or authorized for use as a critical component in life support systems or any FDA Class 3 medical devices or medical devices with a same or similar classification in a foreign jurisdiction or any devices intended for implantation in the human body. Should Buyer purchase or use **onsemi** products for any such unintended or unauthorized application, Buyer shall indemnify and hold **onsemi** and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that **onsemi** was negligent regarding the design or manufacture of the part. **onsemi** is an Equal Opportunity/Affirmative Action Employer. This literature is subject to all applicable copyright laws and is not for resale in any manner.

PUBLICATION ORDERING INFORMATION

LITERATURE FULFILLMENT:
Email Requests to: orderlit@onsemi.com

onsemi Website: www.onsemi.com

TECHNICAL SUPPORT
North American Technical Support:
Voice Mail: 1 800−282−9855 Toll Free USA/Canada
Phone: 011 421 33 790 2910

Europe, Middle East and Africa Technical Support:
Phone: 00421 33 790 2910
For additional information, please contact your local Sales Representative

◊