Skywire® CAT4 LTE Modem Linux Networking Guide

NimbeLink Corp

Updated: December 2020

Table of Contents

1. Introduction

1.1 Scope

This document serves as a networking guide for NL-SW-LTE-TC4NAG Skywire modems and Linux host systems running Debian-based operating systems.

1.2 Overview

This application note covers four networking protocols for Linux environments: PPP, CDC_ECM, MBIM, and QMI.

All protocols allow Skywire modems to provide an Internet connection when connected over USB. The Internet connection provided by the modem behaves similarly to an Ethernet connection, so this networking technique is often called "Ethernet over USB".

For connections over UART, PPP can be used.

Section 2 covers preliminary steps that should be performed, including: checking for package updates, downloading new packages, powering on the modem, and verifying modem enumeration on the Linux PC. Follow the instructions in this section before trying the following.

Linux users can choose from PPP, CDC_ECM, MBIM, and QMI protocols when establishing an Ethernet over USB connection:

- Section 3: Point-to-Point Protocol (PPP)
- Section 4: Communications Device Class Ethernet Control Model (CDC_ECM)
- Section 5: Mobile Broadband Interface Model (MBIM)
- Section 6: Qualcomm Mobile Station Modems (MSM) Interface (QMI)

Note: *CDC_ECM, MBIM, and QMI generally have higher download and upload speeds than PPP.*

The networking methods described in this guide have been tested on the following Linux distributions:

- Ubuntu 18.04 LTS
- Ubuntu 20.04 LTS
- Debian 9.5 2018-10-07

1.3 Orderable Part Numbers

Orderable Device	Cellular Carrier	Network Type
NL-SW-LTE-TC4NAG	AT&T, Verizon	LTE, 3G
NL-SWDK	Any	Any

2. Preliminary Setup Procedure

2.1 Linux PC Setup

2.1.1 Notes

The commands in this guide are for the Debian and Ubuntu listed in Section 1.2. If you are using a different Linux distribution, please modify the commands as needed.

2.1.2 Elevate to Root

To make the process easier, it is best to elevate to root before continuing with this guide. To do so, open a Linux terminal, and issue the following command:

```
sudo -i
```

2.1.3 Check for Package Updates

Ensure that host system has the most recent versions of installed packages by issuing the following command:

```
apt update && apt upgrade
```

This command will check for package updates, and then perform any updates it finds.

2.1.4 Install Required Packages

Additional packages may need to be installed on the Linux PC, depending on the networking protocol(s) that will be tested:

- **PPP**
    - pppd - Point-to-Point Protocol Daemon.
        - Install with: `apt install ppp`

        - The "ppp" package adds a background process that handles PPP traffic alongside the kernel's PPP driver. This is known as a PPP daemon (PPPd).
        - For more information, see the following man page: https://manpages.debian.org/testing/ppp/pon.1.en.html

- **CDC_ECM**
    - No additional packages required.
- **MBIM**
    - Install with: **`sudo apt install libmbim-utils`**
    - For more information, see the following man page: https://manpages.debian.org/unstable/libmbim-utils/mbimcli.1.en.html
- **QMI**
    - Install with: **`sudo apt install libqmi-utils`**
    - For more information, see the following man page: https://manpages.debian.org/unstable/libqmi-utils/qmicli.1.en.html

2.1.5 Install Optional Packages

Depending on the chosen method of communication between the Linux PC and the Skywire, additional package installations may be required:

- picocom - Minimal Dumb-Terminal Emulation Program
    - Install with: **`apt install picocom`**
    - NimbeLink recommends using picocom to handle serial and USB communication between the modem and the Linux host PC.
    - For more information, see the following man page:

      https://linux.die.net/man/8/picocom
    - Any similar package, like minicom, can be used instead of picocom, if desired.

2.1.6 Linux Namespaces

In addition to the standard procedures for PPP and CDC_ECM, this application note has alternate instructions that make use of Linux Namespaces.

A namespace is a container within the operating system that allows for certain processes or resources to be isolated from the rest of the system. Specifically, a Network namespace can be used to isolate the Ethernet-over-USB connection from any other network interfaces that may be present on the host PC.

This approach is helpful while testing an implementation of the Skywire, and can be advantageous for designers that use SSH or Telnet to communicate with the Linux PC. Since the standard PPP and CDC_ECM procedures involve disabling network interfaces, the Linux PC loses its ability to communicate over Ethernet. By isolating the Ethernet over USB connection using a Linux Namespace, the problem of having to take down the Ethernet interface on the Linux PC is avoided during testing.

Note: *It is possible to leave network interfaces enabled on the Linux PC, while ensuring that the cellular data connection is the primary source of Internet connectivity. This involves replacing the default route in the kernel's IP routing table with the PPP, CDC_ECM, QMI, or MBIM connection. However, these steps are not covered by this application note.*

The alternate methods involving Linux Namespaces are meant for testing and prototyping use only. It is not recommended to utilize Linux Namespaces in the production implementation of an Ethernet over USB connection. Instead, follow the standard procedure(s) when designing an implementation for production devices.

2.2 Skywire Modem Setup

2.2.1 Prepare the Hardware

Ensure that the modem has an activated SIM card inserted into the SIM slot, or that the modem's soldered-down SIM is active.

Power on the modem and allow it to boot up. Using a Linux terminal or another preferred method, open a serial or USB connection to the Skywire.

- ***SWDK Users:*** *Refer to the SWDK user manual for detailed setup instructions:*

  https://nimbelink.com/Documentation/Development_Kits/NL-SWDK/30005_NL-SWDK_UserManual.pdf

- *Ensure that the active firmware image corresponds to the SIM card in use. Section 3.9 of the NL-SW-LTE-TC4NAG datasheet contains information regarding the dual image firmware:*

  https://nimbelink.com/Documentation/Skywire/4G_LTE_Cat_4_Telit/1002147_NL-SW-LTE-TC4NAG_Datasheet.pdf

2.2.2 Configure the Modem's USB Composition

Depending on the networking protocol being used, the Skywire's USB composition may need to be changed. That is, the modem must be configured so that the proper USB ports are available to the Linux host PC.

Open up a terminal emulator program on the Linux PC (picocom, PuTTY, etc.), and connect to the modem's USB or serial interface. Issue one of the following AT commands, depending on the networking protocol that has been chosen for testing:

- **PPP and QMI**
  - Choose the firmware-default USB composition:

    ```
    AT#USBCFG=0
    ```

  - This composition enables the following USB ports:

    ```
    DIAG + ADB + RMnet + NMEA + MODEM + MODEM + SAP
    ```

  - The USB Product ID (PID) of this composition is:

    ```
    0x1201
    ```

- **CDC_ECM**
  - Choose the following USB interface configuration:

    ```
    AT#USBCFG=4
    ```
  - The composition of this configuration is:

    ```
    DIAG + ADB + ECM + NMEA + MODEM + MODEM + SAP
    ```
  - The USB Product ID (PID) of this composition is:

    ```
    0x1206
    ```
- **MBIM**
  - Choose the following USB interface configuration:

    ```
    AT#USBCFG=2
    ```
  - The composition of this configuration is:

    ```
    DIAG + ADB + MBIM + NMEA + MODEM + MODEM + SAP
    ```
  - The USB Product ID (PID) of this composition is:

    ```
    0x1204
    ```

Note 1: *The USB Vendor ID for all Telit-based Skywire modems is* **`0x1b7c`**, regardless of the current USB composition.

Note 2: *See the AT command manual for the NL-SW-LTE-TC4NAG for further information regarding the "**`AT#USBCFG`**" command:*

https://nimbelink.com/Documentation/Skywire/4G_LTE_Cat_4_Telit/Telit_LE910Cx_AT_Commands_Reference_Guide_r2.pdf

The table below summarizes the relevant information for each of the USB interface compositions described above:

Protocol	Required USB Composition	USB VID	USB PID
PPP/QMI	`#USBCFG: 0`	0x1b7c	0x1201
CDC_ECM	`#USBCFG: 4`	0x1b7c	0x1206
MBIM	`#USBCFG: 2`	0x1bc7	0x1204

Once the USB composition has been changed, the modem will shut down and then reboot with the new composition. After the modem reboots, verify that the new configuration was applied correctly.

2.2.3 Verify Modem Enumeration

After changing the USB composition, verify that the modem has properly enumerated on the Linux PC. Type the following command into the Linux terminal:

```
lsusb
```

If the Skywire's USB interface has properly enumerated on the Linux PC, there should be a device listed that looks something like:

```
Bus 001 Device 02: ID 1bc7:[PID] Telit Wireless Solutions
```

Where "**[PID]**" is replaced with the PID that corresponds to the active USB interface composition. See the table above for possible PID's.

If a line similar to the one above appears, then the modem has properly enumerated on the Linux PC.

If a USB device does not appear, verify that the modem is powered on and that the USB interface is connected to the Linux PC. Then, retry the "**lsusb**" command.

If the modem still has not enumerated as expected on the Linux PC, try the troubleshooting step in the next section.

2.2.4 Load the “option” Driver (If Needed)

On older versions of Ubuntu and other Linux distributions, newer Skywire modems will sometimes fail to enumerate fully, if at all.

If the modem does not enumerate on the host PC, or if some of the modem's USB ports are missing from the Linux PC, the "option" driver must be loaded for the modem. Issue the following commands:

```
modprobe option

echo 1bc7 [PID] > /sys/bus/usb-serial/drivers/option1/new_id
```

Be sure to replace "**[PID]**" with the PID that corresponds to the active USB composition for the Skywire. See the table above for possible PID's.

In the case of an NL-SW-LTE-TC4NAG Skywire with the default USB composition, the above command would be:

```
echo 1bc7 1201 > /sys/bus/usb-serial/drivers/option1/new_id
```

After issuing the above command, the Skywire's USB interface should now fully enumerate on the Linux PC.

At this point, the Linux PC and the Skywire are ready for the Ethernet over USB connection. Section 3 contains instructions for PPP, Section 4 contains instructions for CDC_ECM, Section 5 contains instructions for MBIM, and Section 6 contains instructions for QMI.

Note: *If the modem still does not enumerate, then it is possible that the USB interface is damaged on the modem. For further assistance, please contact the following email:*

product.support@nimbelink.com

3. Point-to-Point Protocol (PPP)

3.1 Overview

Point-to-Point Protocol (PPP) is a communications protocol used to directly connect two devices together without any other networking devices placed in between. PPP is a data link layer protocol, and is designed to work with multiple network layer protocols, such as Internet Protocol (IP).

Section 3 describes how to set up PPP on an NL-SW-LTE-TC4NAG Skywire modem and a Linux host system like a PC, or a single-board computer (BeagleBone, Raspberry Pi, etc.).

Note: *These instructions assume that the Linux host PC and the Skywire have both been set up according to Section 2, and that the "pppd" package has been installed on the PC.*

3.2 PPP Scripts

3.2.1 NimbeLink PPP Script GitHub Repository

NimbeLink has a library of PPP scripts available for customer use at the following link:

https://github.com/NimbeLink/skywire-ppp-scripts

NimbeLink recommends downloading this repository, and following the instructions on the GitHub page to set up the scripts.

To clone the repository, navigate to any desired directory, and enter the following command into the Linux terminal:

```
git clone https://github.com/NimbeLink/skywire-ppp-scripts.git
```

Alternatively, the contents of the PPP scripts can be copied directly from an Internet browser into the proper location on the Linux PC's filesystem. However, it is recommended to simply clone the repository.

3.2.2 Create PPP Scripts

As root user, navigate to the "**`/etc/ppp/peers`**" directory on the Linux host PC. This directory is where all of the PPP scripts for Skywire modems should be placed.

Next, identify the pair of PPP scripts that correspond to the carrier of the SIM card in use, and then copy those scripts to the "**`/etc/ppp/peers`**" directory.

The commands below demonstrate this operation, assuming that the PPP script repository was cloned into the home directory of the root user. Change the path to the "**`skywire-ppp-scripts`**" repository in the command below, if applicable:

Verizon Users:

```
cp ~/skywire-ppp-scripts/vzw-TC4NAG* /etc/ppp/peers
```

AT&T Users:

```
cp ~/skywire-ppp-scripts/att-TC4NAG* /etc/ppp/peers
```

To Copy Both AT&T and Verizon Scripts:

```
cp ~/skywire-ppp-scripts/*TC4NAG* /etc/ppp/peers
```

If the copying operation was done correctly, the NL-SW-LTE-TC4NAG PPP scripts should now be in the "**`/etc/ppp/peers`**" directory.

Verify this by comparing the contents of the "**`/etc/ppp/peers`**" with the list of required PPP scripts below:

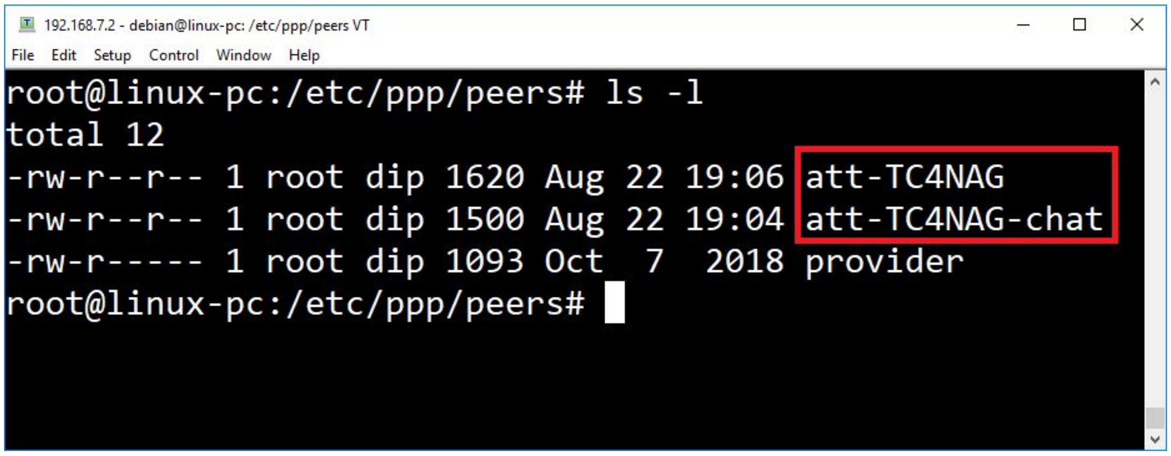
Required Verizon PPP Scripts:

- "**`vzw-TC4NAG-chat`**"
- "**`vzw-TC4NAG`**"

Required AT&T PPP Scripts:

- "**`att-TC4NAG-chat`**"
- "**`att-TC4NAG`**"

For example, the image below depicts the contents of the "**`/etc/ppp/peers`**" directory with the two required AT&T scripts:


192.168.7.2 - debian@linux-pc: /etc/ppp/peers VT
File Edit Setup Control Window Help

```
root@linux-pc:/etc/ppp/peers# ls -l
total 12
-rw-r--r-- 1 root dip 1620 Aug 22 19:06 att-TC4NAG
-rw-r--r-- 1 root dip 1500 Aug 22 19:04 att-TC4NAG-chat
-rw-r----- 1 root dip 1093 Oct  7  2018 provider
root@linux-pc:/etc/ppp/peers#
```

3.2.3 Edit the *"x-TC4NAG-chat"* Script

Before using the scripts, the "**`x-TC4NAG-chat`**" file must be edited to add the proper APN for the cellular connection. Replace "**`x`**" in the filename with either "**`att`**" or "**`vzw`**", depending on the SIM card being used.

Open the "**`x-TC4NAG-chat`**" file with a preferred text editor, and find the line that says:

AT&T Users:

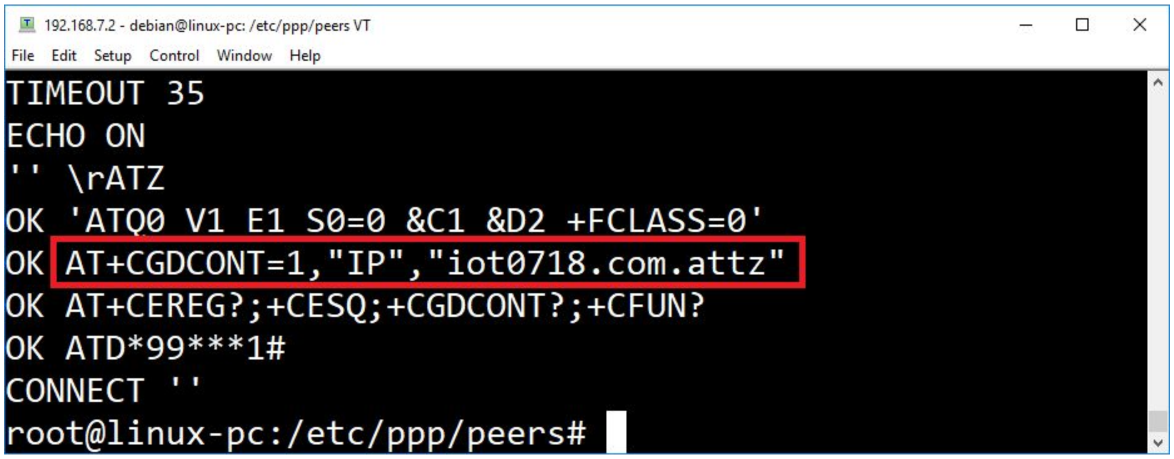
- **`OK AT+CGDCONT=1,"IP","[apn]"`**

Verizon Users:

- **`OK AT+CGDCONT=3,"IP","[apn]"`**

Replace "**`[apn]`**" with the proper APN for the chosen SIM card and cellular carrier.

For example, the image below depicts an edited "**`att-TC4NAG-chat`**" file with the proper APN set for an AT&T SIM card purchased through NimbeLink:


192.168.7.2 - debian@linux-pc: /etc/ppp/peers VT
File Edit Setup Control Window Help

```
TIMEOUT 35
ECHO ON
'' \rATZ
OK 'ATQ0 V1 E1 S0=0 &C1 &D2 +FCLASS=0'
OK AT+CGDCONT=1,"IP","iot0718.com.attz"
OK AT+CEREG?;+CESQ;+CGDCONT?;+CFUN?
OK ATD*99***1#
CONNECT ''
root@linux-pc:/etc/ppp/peers#
```

3.2.4 Edit the *"x-TC4NAG"* Script

Next, the "**`x-TC4NAG`**" file may need to be updated to reflect the proper port for the PPP session. Replace "**`x`**" in the filename with either "**`att`**" or "**`vzw`**", depending on the SIM card being used.

Open this file with a preferred text editor, and change the line that says "**`/dev/ttyUSB3`**" to reflect the port to which the modem enumerated (if necessary).

For instance, if the modem enumerated to the "**`/dev/ttyACM0`**" port, this port should be specified in the "**`x-TC4NAG`**" file.

3.3 Choose a PPP Implementation

After the PPP scripts have been copied and edited as described above, they are ready to be used for the PPP session. Choose one of the following sections to continue:

Section 3.4: Standard PPP Procedure

- **This is the standard PPP setup procedure that should be used for production implementations of PPP.**
- This method also works well for testing, provided that the user is not using SSH or Telnet to communicate with the Linux PC over Ethernet.

Section 3.5: Alternate PPP Procedure for Testing Purposes

- **This is an alternate method for setting up PPP meant for testing only. Do not use this method for production implementations of PPP.**
- This method is useful for readers who use SSH or Telnet to communicate with the Linux PC over Ethernet, because it won't sever that communication path like the standard PPP procedure does.

Additionally, Section 3.6 contains troubleshooting tips that may be useful when attempting to start the PPP session.

3.4 Standard PPP Procedure

3.4.1 Take down Network Interfaces

Note: *This step will sever the communication path over Ethernet with the Linux PC.*

Issue the following command in the Linux terminal:

```
ifconfig
```

If needed, install with **`sudo apt install net-tools`**.

This command lists all currently active network interfaces. Take note of each entry that is populated in the response to the above command.

Disable each interface that may provide an Internet connection to the Linux PC. Issue the following command to take each interface down:

```
ifconfig [interface] down
```

Replace "**`[interface]`**" with the name of the interface that is to be disabled. Common network interfaces are: **`eth#, enp#s#, wlan#, wwan#, etc.`**

Once the network interfaces have been disabled, check the PC's Internet connectivity using the Linux terminal:

```
ping 8.8.8.8 -c 5
```

This will cause the Linux PC to attempt to ping Google's public DNS server five times. If all of the relevant network interfaces have been disabled, the ping attempt will fail.

If the ping attempt succeeds, then verify that all relevant network interfaces have been disabled with the "**`ifconfig [interface] down`**" command.

Once all sources of Internet connectivity have been disabled, the PPP session can be tested properly.

3.4.2 Start the PPP Session

Start the PPP session with the following command:

```
pon [ppp script]
```

Replace "**`[ppp script]`**" with one of the following filenames, depending on the carrier of the chosen SIM card:

Verizon Users: "**`vzw-TC4NAG`**"

AT&T Users: "**`att-TC4NAG`**"

This command will start the pppd daemon, which will configure the PPP connection, and set it as the default network connection.

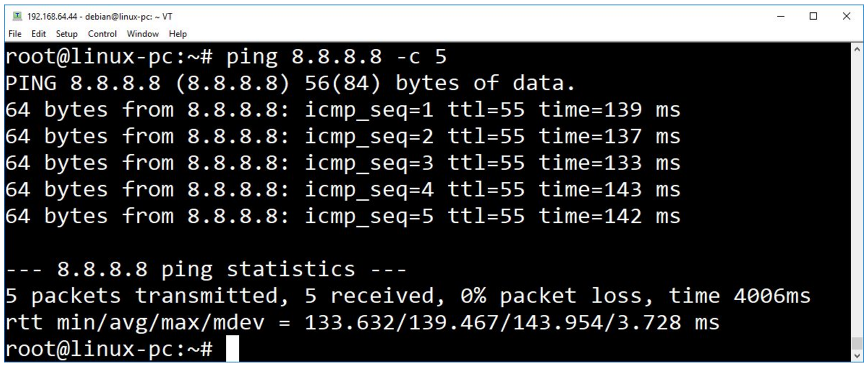
Note: *Ensure that the modem does not have an active PDP context when starting the PPP script. This will cause the PPP script to encounter an error and freeze up.*

3.4.3 Test the PPP Session

To test that the PPP connection is working properly, issue the ping command once more:

```
ping 8.8.8.8 -c 5
```

If the PPP session was successful, the response should look something like:


192.168.64.44 - debian@linux-pc: ~ VT
File Edit Setup Control Window Help

```
root@linux-pc:~# ping 8.8.8.8 -c 5
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=55 time=139 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=55 time=137 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=55 time=133 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=55 time=143 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=55 time=142 ms

--- 8.8.8.8 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 133.632/139.467/143.954/3.728 ms
root@linux-pc:~#
```

3.4.4 Close the PPP Connection

To close the PPP connection, simply issue this command:

```
poff
```

It is a good idea to close a PPP session whenever it is no longer needed. This will prevent accidental usage of cellular data.

Note: *This concludes the standard implementation of PPP for NL-SW-LTE-TC4NAG Skywire modems.*

3.5 Alternate Procedure for Testing PPP

This section describes a different PPP setup process that allows the user to properly test the PPP connection without taking down the Ethernet interface on the Linux PC.

The demonstration of the alternate PPP setup is meant for testing and prototyping purposes only. It is not recommended to implement PPP using Linux namespaces for final production devices. If a communication path over Ethernet is required for production devices, additional Linux networking steps (beyond those covered in this guide) may be required.

3.5.1 Create a Network Namespace

Create a Network namespace called "**`ppp-testing`**" on the Linux host PC, type the following command:

```
ip netns add ppp-testing
```

If desired, replace "**`ppp-testing`**" with any descriptive name for the new namespace.

This new namespace is completely isolated from any of the current network interfaces present on the Linux PC. To demonstrate this, issue the following command:

```
ip netns exec ppp-testing ifconfig
```

No network interfaces should appear as a result of the above command.

3.5.2 Start the PPP Session

Next, start the PPP session within the namespace with the following command:

```
ip netns exec ppp-testing pon <ppp script>
```

Replace "`<ppp script>`" with one of the filenames below:

Verizon Users:

- "**`vzw-TC4NAG`**"

AT&T Users:

- "**`att-TC4NAG`**"

This command will start the pppd daemon, which will configure the PPP connection, and set it as the default network connection for the namespace.

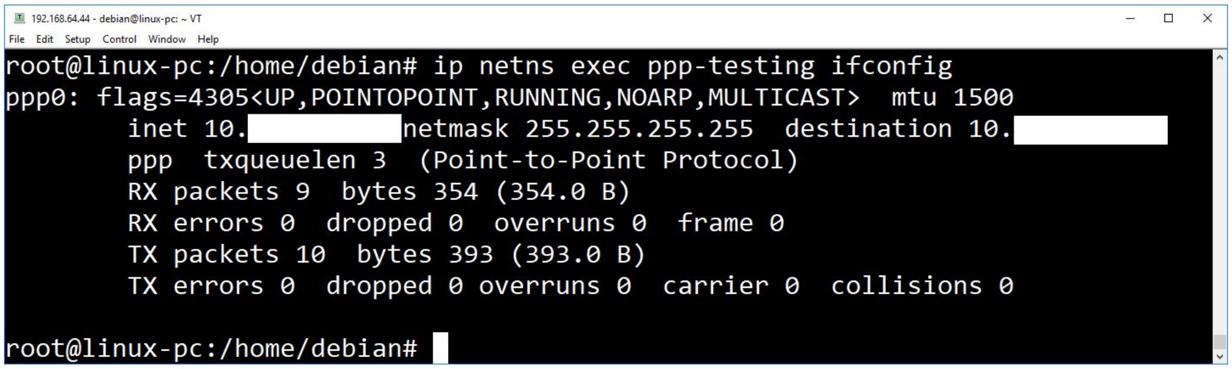
Note: *Ensure that the modem does not have an active PDP context when starting the PPP script. This will cause the PPP script to encounter an error and freeze up.*

3.5.3 Test the PPP Connection

There should now be a "**`ppp0`**" interface contained within the "**`ppp-testing`**" network namespace. To check this, issue the following command:

```
ip netns exec ppp-testing ifconfig
```

The terminal should respond with something similar to:


192.168.64.44 - debian@linux-pc: ~ VT
File Edit Setup Control Window Help

```
root@linux-pc:/home/debian# ip netns exec ppp-testing ifconfig
ppp0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 10.[redacted]netmask 255.255.255.255  destination 10.[redacted]
        ppp  txqueuelen 3  (Point-to-Point Protocol)
        RX packets 9  bytes 354 (354.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 10  bytes 393 (393.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@linux-pc:/home/debian# 
```

The "**`ppp0`**" interface should be the only active network interface listed for the "**`ppp-testing`**" network namespace, as seen in the image above.

For the sake of demonstration, query the network interfaces for the Linux PC (outside of the "**`ppp-testing`**" network namespace) using the following command:

```
ifconfig
```

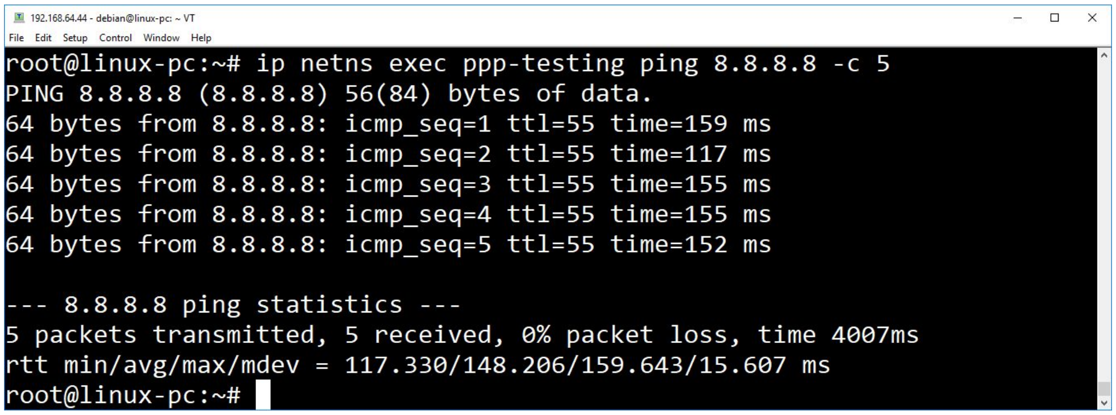
The "**ppp0**" network interface should not be listed as a network interface for the Linux PC in the response to the above command.

This is because the PPP session was run in the "container" of the Linux namespace that was created for the testing, and is not visible to anything outside of the "**ppp-testing**" network namespace.

To test that the PPP connection is functioning properly, issue the following command:

```
ip netns exec ppp-testing ping 8.8.8.8 -c 5
```

This command will ping Google's public DNS server in order to test the connection. The response to the ping command should look something like the image below:


192.168.64.44 - debian@linux-pc: ~ VT
File Edit Setup Control Window Help

```
root@linux-pc:~# ip netns exec ppp-testing ping 8.8.8.8 -c 5
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=55 time=159 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=55 time=117 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=55 time=155 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=55 time=155 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=55 time=152 ms

--- 8.8.8.8 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 117.330/148.206/159.643/15.607 ms
root@linux-pc:~#
```

If the ping command produces results similar to the image above, then it is safe to say that the Internet connection has been established.

3.5.4 Close the PPP Connection

To close the PPP connection, simply issue this command:

```
ip netns exec ppp-testing poff
```

It is a good idea to close a PPP session whenever it is no longer needed. This will prevent accidental usage of cellular data.

3.5.5 Delete the Linux Namespace

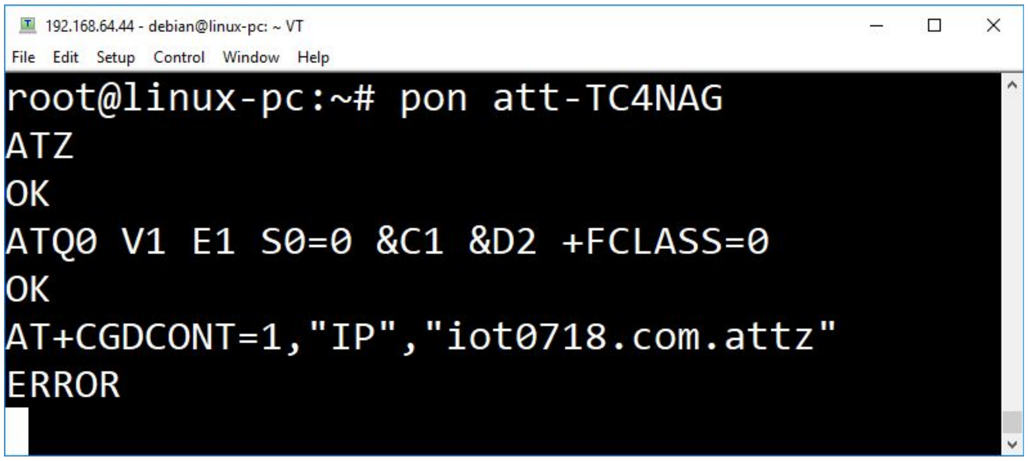
To delete the Linux namespace, issue the following command:

```
ip netns del ppp-testing
```

The Linux namespace will also be deleted upon reboot of the Linux PC, so it is not entirely necessary to delete it after testing has concluded.

3.6 PPP Troubleshooting

If the PPP scripts get hung up:


192.168.64.44 - debian@linux-pc: ~ VT
File Edit Setup Control Window Help

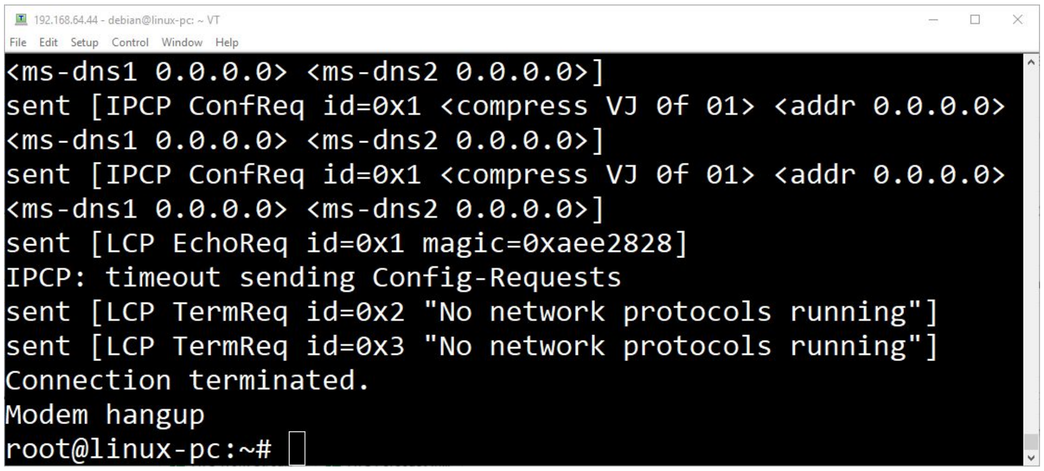
```
root@linux-pc:~# pon att-TC4NAG
ATZ
OK
ATQ0 V1 E1 S0=0 &C1 &D2 +FCLASS=0
OK
AT+CGDCONT=1,"IP","iot0718.com.attz"
ERROR
```

1. If the Skywire has an active PDP context, the PPP scripts will freeze like this.
    a. To fix this, open a serial or USB connection to the modem, and instruct the modem to reboot:

    ```
    AT#REBOOT
    ```

2. The modem should be freshly powered up, or rebooted each time a PPP session is started.

If the PPP scripts timeout:


192.168.64.44 - debian@linux-pc: ~ VT
File Edit Setup Control Window Help

```
<ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
sent [IPCP ConfReq id=0x1 <compress VJ 0f 01> <addr 0.0.0.0>
<ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
sent [IPCP ConfReq id=0x1 <compress VJ 0f 01> <addr 0.0.0.0>
<ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
sent [LCP EchoReq id=0x1 magic=0xaee2828]
IPCP: timeout sending Config-Requests
sent [LCP TermReq id=0x2 "No network protocols running"]
sent [LCP TermReq id=0x3 "No network protocols running"]
Connection terminated.
Modem hangup
root@linux-pc:~#
```

1. This type of error is generally caused by attempting to start the PPP session too soon after the Skywire has booted.
    a. To fix this, wait 5-10 seconds after the modem is responsive before starting the PPP scripts.

4. CDC_ECM

4.1 Overview

The Communication Device Class Ethernet Control Model (CDC_ECM) is a USB communication protocol developed by the USB Implementers Forum. This protocol enables the sending and receiving of IEEE 802.3 Ethernet frames over a USB bus.

For more information regarding this communication protocol, please refer to the CDC specification at the following link:

https://www.usb.org/document-library/cdc-subclass-specification-ethernet-emulation-model-devices-10

Section 4 describes how to set up CDC_ECM on an NL-SW-LTE-TC4NAG Skywire modem and a Linux host system like a PC, or a single-board computer (BeagleBone, Raspberry Pi, etc.).

Note: *These instructions assume that the Linux host PC and the Skywire have both been set up according to Section 2. No additional packages are required to use CDC_ECM.*

4.2 Preliminary Skywire Configuration

4.2.1 Verify USB Composition

In order for CDC_ECM to work with the Skywire, the USB composition must be set correctly on the modem.

The USB composition should have been set correctly in Section 2.2.2, but it is a good idea to verify it either way. Issue the following command:

```
AT#USBCFG?
```

The modem should respond with the following:

```
#USBCFG: 4
```

If the USB composition is set correctly, proceed to Section 4.2.2. Otherwise, issue the following command to obtain the proper setting:

```
AT#USBCFG=4
```

The modem will then power down, and reboot with the correct composition. Once the AT command parser is responsive, issue the "**AT#USBCFG?**" command again to verify that the correct setting is selected.

4.2.2 Configure a PDP Context

A PDP context must be configured on the modem before starting the CDC_ECM session. To do so, issue one of the following AT commands, depending on the SIM card in use, and the currently active firmware image:

Verizon Users:

```
AT+CGDCONT=3,"IPV4V6","[APN]"
```

AT&T Users:

```
AT+CGDCONT=1,"IPV4V6","[APN]"
```

Replace "`[APN]`" with the APN for the currently selected SIM card.

To make sure the PDP context was configured properly, query the PDP context and verify that the APN is correct:

```
AT+CGDCONT?
```

4.2.3 Activate CDC_ECM

The final step during the Skywire configuration process is to enable CDC_ECM on the modem. To do so, issue the following command, depending on the SIM card in use, and the currently active firmware image:

Verizon Users:

```
AT#ECM=3,0
```

AT&T Users:

```
AT#ECM=1,0
```

The first argument to this command specifies the PDP context to use for the CDC_ECM session, and the second argument is the "Device ID", which is firmware-limited to a value of "`0`".

The modem will take a second or two to process this command, but will eventually respond with "`OK`". After this command has successfully been entered, the modem is ready for the CDC_ECM connection.

If the modem returns an error when issuing this command, refer to Section 4.6 for further troubleshooting information.

4.3 Choose a CDC_ECM Implementation

After the preliminary Skywire configuration steps have been completed, the modem is ready for the CDC_ECM session. Choose one of the following sections to continue:

Section 4.4: Standard CDC_ECM Procedure

- **This is the standard CDC_ECM setup procedure that should be used for production implementations of CDC_ECM.**
- This method also works well for testing, provided that the user is not using SSH or Telnet to communicate with the Linux PC over Ethernet.

Section 4.5: Alternate CDC_ECM Procedure for Testing Purposes

- **This is an alternate method for setting up CDC_ECM meant for testing only. Do not use this method for production implementations of CDC_ECM.**
- This method is useful for readers who use SSH or Telnet to communicate with the Linux PC over Ethernet, because it won't sever that communication path like the standard CDC_ECM procedure does.

Additionally, Section 4.6 contains troubleshooting tips that may be useful when attempting to start the CDC_ECM session.

4.4 Standard CDC_ECM Procedure

4.4.1 Take down Network Interfaces

Note: *This step will sever the communication path over Ethernet with the Linux PC.*

Issue the following command in the Linux terminal:

```
ifconfig
```

This command will list all currently active network interfaces. Take note of each of the entries that are populated in the response to the above command.

Bring down each interface that may be providing an Internet connection to the Linux PC. Issue the following command to take each interface down:

```
ifconfig [interface] down
```

Replace "**`[interface]`**" with the name of the interface that is to be disabled. Common network interfaces are: **`eth#, enp#s#, wlan#, etc.`**

Once the network interfaces have been disabled, check the PC's Internet connectivity:

```
ping 8.8.8.8 -c 5
```

This will cause the Linux PC to attempt to ping Google's public DNS server five times.

If all of the relevant network interfaces have been disabled, the ping attempt will fail.

If the ping attempt succeeds, verify that all relevant network interfaces have been disabled with the "`ifconfig [interface] down`" command.

Once all sources of Internet connectivity have been disabled, the CDC_ECM session can be tested properly.

4.4.2 Identify the wwan[#] Interface

In the current USB composition setting, the modem's ECM USB port should have enumerated as a Wireless Wide Area Network (WWAN) device on the Linux PC. To check this, issue the following command:

```
ifconfig -a
```

This will query the network interfaces on the Linux PC, even those that are currently disabled. The response to this command should list several network interfaces, including those that were disabled in Section 4.4.1.

Among those disabled network interfaces should be a "`wwan[#]`" interface, typically "`wwan0`". This is modem's USB ECM interface.

4.4.3 Enable the wwan# Interface

First, flush any existing IP address information that may be associated with the "`wwan[#]`" network interface. Send the following command to the Linux PC after replacing `[#]` with the proper number::

```
ip addr flush dev wwan[#]
```

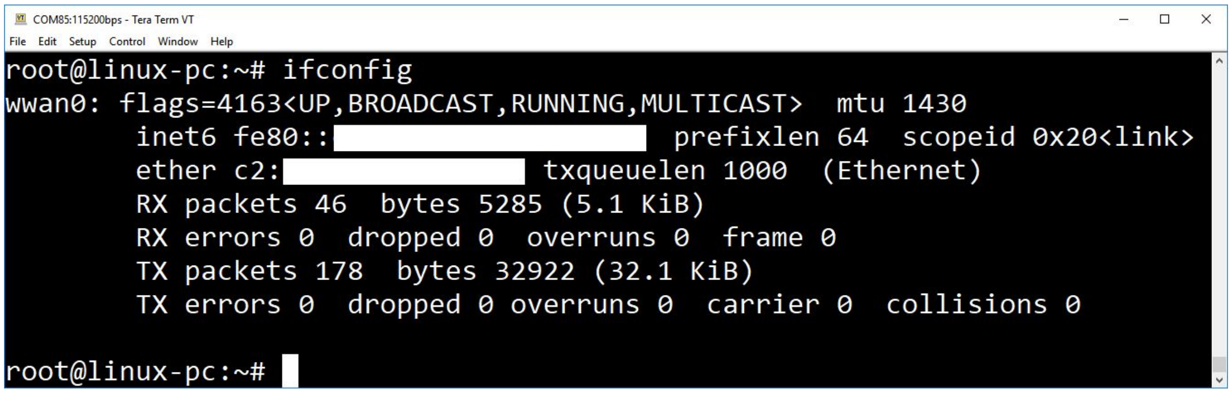
Then, enable the "`wwan[#]`" interface with the following command:

```
ifconfig wwan[#] up
```

Afterwords, issue this command to verify that the "`wwan[#]`" network interface was enabled correctly:

```
ifconfig
```

The "`wwan[#]`" interface should be listed as an active network interface. See the image below for reference:


COM85:115200bps - Tera Term VT
File Edit Setup Control Window Help

```
root@linux-pc:~# ifconfig
wwan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1430
        inet6 fe80::[redacted]  prefixlen 64  scopeid 0x20<link>
        ether c2:[redacted]  txqueuelen 1000  (Ethernet)
        RX packets 46  bytes 5285 (5.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 178  bytes 32922 (32.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@linux-pc:~#
```

4.4.4 Dynamically Configure IP Address of wwan[#] Interface

The final step is to dynamically configure the IP Address of the "**`wwan[#]`**" interface using "dhclient". The link below points to the manpage for this tool:

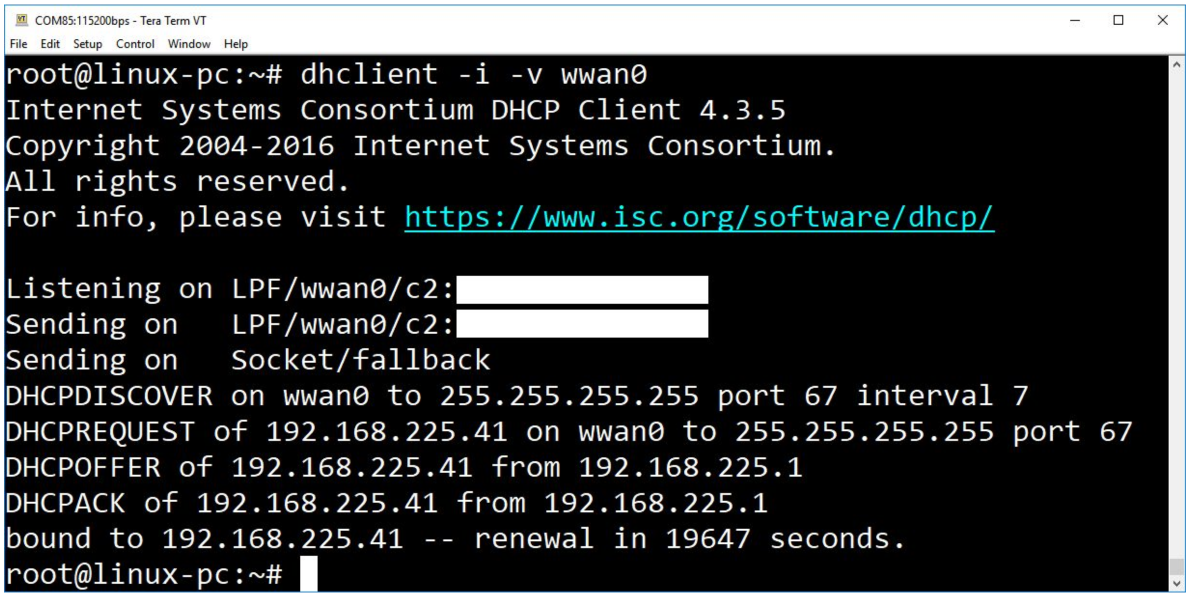
https://linux.die.net/man/8/dhclient

This tool allows the Linux PC to automatically configure a private IP address for the "**`wwan[#]`**" network interface using a DHCP server that is provided by the cellular module.

To do so, issue the following command to the Linux terminal:

```
dhclient -i -v wwan[#]
```

The response the this command should be something like:


COM85:115200bps - Tera Term VT
File Edit Setup Control Window Help

```
root@linux-pc:~# dhclient -i -v wwan0
Internet Systems Consortium DHCP Client 4.3.5
Copyright 2004-2016 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/wwan0/c2:[redacted]
Sending on   LPF/wwan0/c2:[redacted]
Sending on   Socket/fallback
DHCPDISCOVER on wwan0 to 255.255.255.255 port 67 interval 7
DHCPREQUEST of 192.168.225.41 on wwan0 to 255.255.255.255 port 67
DHCPOFFER of 192.168.225.41 from 192.168.225.1
DHCPACK of 192.168.225.41 from 192.168.225.1
bound to 192.168.225.41 -- renewal in 19647 seconds.
root@linux-pc:~#
```

In the above image, it can be seen that the "**`wwan0`**" network interface was assigned a private IP address of "**`192.168.225.41`**" within the subnet generated by the modem.

If "dhclient" was successful in assigning a private IP address to the modem, the CDC_ECM connection should now be up and running.

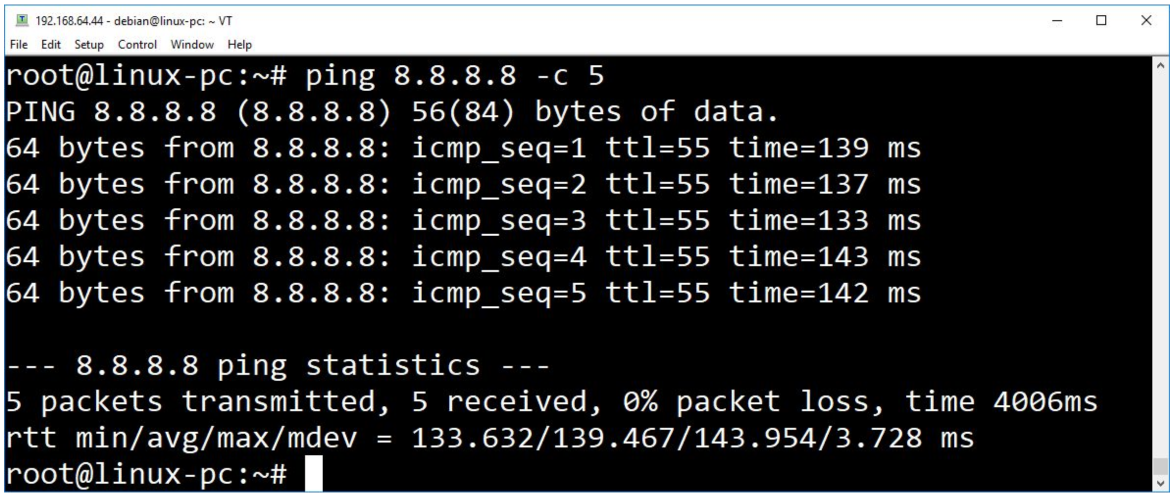
If "dhclient" was not successful in assigning a private IP address, make sure that the modem has the proper APN set in the PDP context, and that "**`AT#ECM`**" has been set properly as per Section 4.2.3.

4.4.5 Test the CDC_ECM Connection

To test that the CDC_ECM connection is working properly, issue the following command:

```
ping 8.8.8.8 -c 5
```

If the CDC_ECM session was successful, the response should look something like:


192.168.64.44 - debian@linux-pc: ~ VT
File Edit Setup Control Window Help

```
root@linux-pc:~# ping 8.8.8.8 -c 5
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=55 time=139 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=55 time=137 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=55 time=133 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=55 time=143 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=55 time=142 ms

--- 8.8.8.8 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 133.632/139.467/143.954/3.728 ms
root@linux-pc:~#
```

4.4.6 Close the CDC_ECM Connection

To stop the CDC_ECM session simply issue the following AT command to the modem:

```
AT#ECMD=0
```

This will turn off CDC_ECM mode on the modem. Next issue the following command to the Linux PC to close the dhclient session:

```
dhclient -r
```

It is a good idea to close a CDC_ECM session whenever it is no longer needed. This will prevent accidental usage of cellular data.

Note: *This concludes the standard implementation of CDC_ECM on an NL-SW-LTE-TC4NAG Skywire.*

4.5 Alternate CDC_ECM Procedure

This section describes a different CDC_ECM setup process that allows the user to properly test the CDC_ECM connection without taking down the Ethernet interface on the Linux PC.

The demonstration of the alternate CDC_ECM setup is meant for testing and prototyping purposes only. It is not recommended to implement CDC_ECM using Linux namespaces for final production devices. If a communication path over Ethernet is required for production devices, additional Linux networking steps (beyond those covered in this guide) may be required.

4.5.1 Create a Linux Namespace

Create a Network namespace called "**`cdc_ecm-testing`**" on the Linux host PC, type the following command:

```
ip netns add cdc_ecm-testing
```

If desired, replace "**`cdc_ecm-testing`**" with any descriptive name for the new namespace.

This new namespace is completely isolated from any of the current network interfaces present on the Linux PC. To demonstrate this, issue the following command:

```
ip netns exec cdc_ecm-testing ifconfig
```

No network interfaces should appear as a result of the above command.

4.5.2 Identify the wwan[#] Interface

In the current USB composition setting, the modem's ECM USB port should have enumerated as a Wireless Wide Area Network (WWAN) device on the Linux PC. To check this, issue the following command:

```
ifconfig -a
```

This will query the network interfaces on the Linux PC, even those that are currently disabled.

Among those disabled network interfaces should be a "**`wwan[#]`**" interface, typically "**`wwan0`**". This is modem's USB ECM interface.

4.5.3 Pass the wwan[#] Interface to the Linux Namespace

The "**`wwan[#]`**" network interface must be passed into the "**`cdc_ecm-testing`**" Linux namespace. To do so, issue the following command after replacing **`[#]`** with the proper number:

```
ip link set wwan[#] netns cdc_ecm-testing
```

To check to see if the above command was successful, issue the following command:

```
ip netns exec cdc_ecm-testing ifconfig -a
```

The "**`wwan[#]`**" inteface should appear in the response to the above command. If it does, then the "**`ip link set ...`**" command was successful.

If the "**`ip link set ...`**" command was not successful, verify that the "**`wwan[#]`**" interface has enumerated on the Linux PC correctly. Also verify that the modem has the correct USB composition, as per the instructions in Section 2.2.2.

4.5.4 Enable the wwan[#] Interface

Flush any existing IP address information that may be associated with the "**`wwan[#]`**" network interface. Send the following command to the Linux PC after replacing **`[#]`** with the proper number:

```
ip netns exec cdc_ecm-testing ip addr flush dev wwan[#]
```

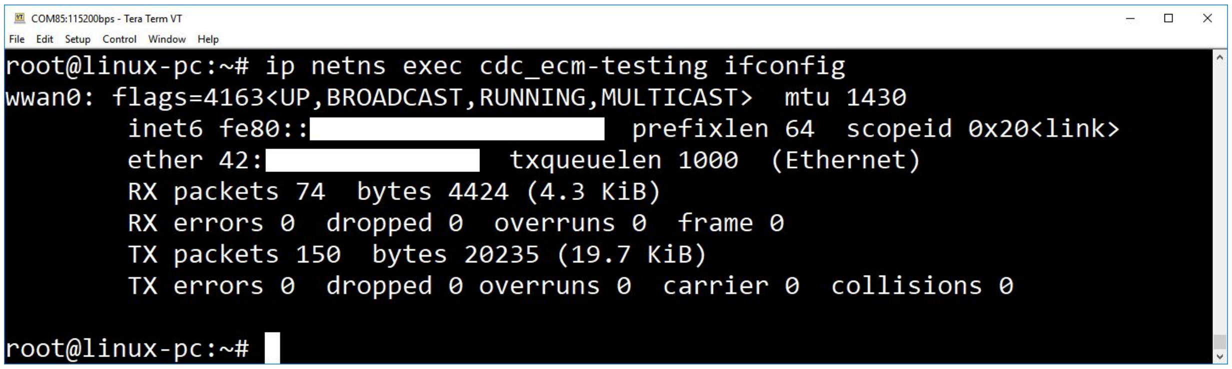
Then, enable the "**`wwan[#]`**" interface with the following command:

```
ip netns exec cdc_ecm-testing ifconfig wwan[#] up
```

Afterwords, issue this command to verify that the "**`wwan[#]`**" network interface was enabled correctly:

```
ip netns exec cdc_ecm-testing ifconfig
```

The "**`wwan[#]`**" interface should be listed as an active network interface. See the image below for reference:


COM85:115200bps - Tera Term VT
File Edit Setup Control Window Help

```
root@linux-pc:~# ip netns exec cdc_ecm-testing ifconfig
wwan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1430
        inet6 fe80::[redacted]  prefixlen 64  scopeid 0x20<link>
        ether 42:[redacted]  txqueuelen 1000  (Ethernet)
        RX packets 74  bytes 4424 (4.3 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 150  bytes 20235 (19.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@linux-pc:~#
```

4.5.5 Dynamically Configure IP Address of wwan[#] Interface

The final step is to dynamically configure the IP Address of the "**`wwan[#]`**" interface using "dhclient". The link below contains the manpage for this tool:

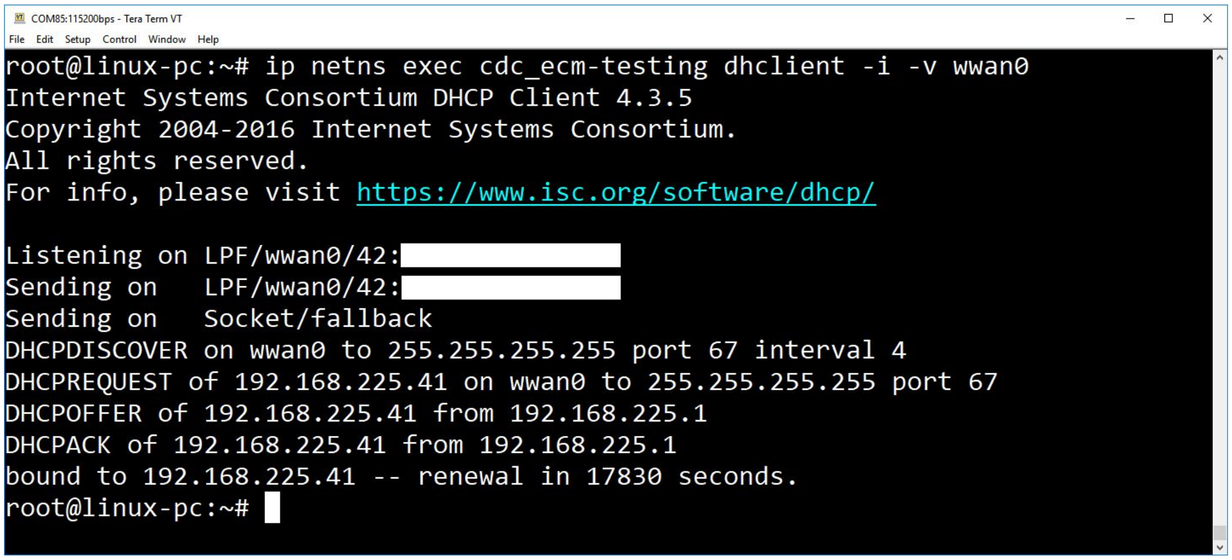
https://linux.die.net/man/8/dhclient

This tool allows the Linux PC to automatically configure a private IP address for the "**`wwan[#]`**" network interface using a DHCP server that is provided by the cellular module.

To do so, issue the following command to the Linux terminal:

```
ip netns exec cdc_ecm-testing dhclient -i -v wwan[#]
```

The response the this command should be something like:


COM85:115200bps - Tera Term VT
File Edit Setup Control Window Help

```
root@linux-pc:~# ip netns exec cdc_ecm-testing dhclient -i -v wwan0
Internet Systems Consortium DHCP Client 4.3.5
Copyright 2004-2016 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/wwan0/42:[redacted]
Sending on   LPF/wwan0/42:[redacted]
Sending on   Socket/fallback
DHCPDISCOVER on wwan0 to 255.255.255.255 port 67 interval 4
DHCPREQUEST of 192.168.225.41 on wwan0 to 255.255.255.255 port 67
DHCPOFFER of 192.168.225.41 from 192.168.225.1
DHCPACK of 192.168.225.41 from 192.168.225.1
bound to 192.168.225.41 -- renewal in 17830 seconds.
root@linux-pc:~#
```

In the above image, we can see that the "**`wwan0`**" network interface was assigned a private IP address of "**`192.168.225.41`**" within the subnet generated by the cellular modem.

If "dhclient" was successful in assigning a private IP address to the modem, the CDC_ECM connection should now be up and running.

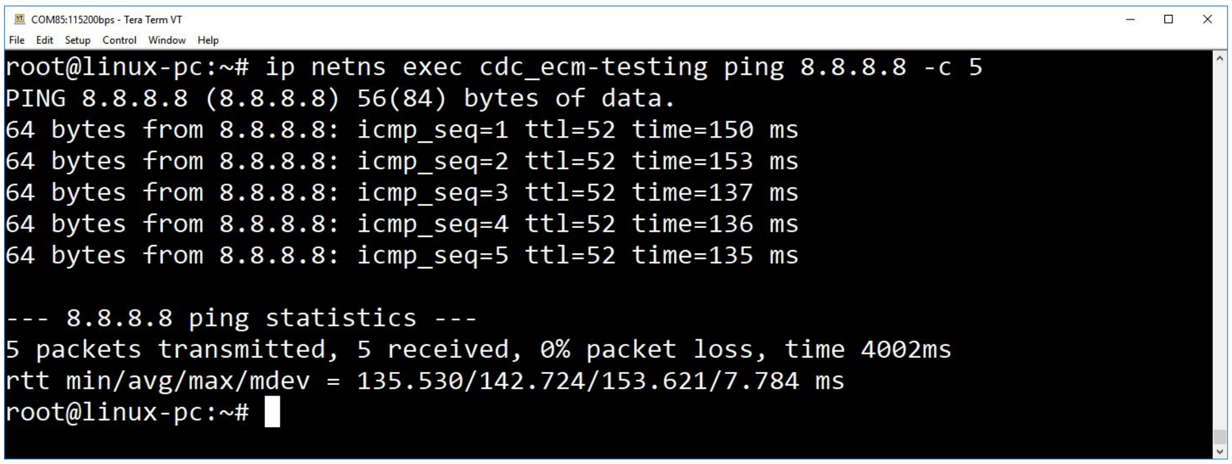
If "dhclient" was not successful in assigning a private IP address, make sure that the modem has the proper APN set in the PDP context, and that "**`AT#ECM`**" has been set properly as per Section 4.2.3.

4.5.6 Test the CDC_ECM Connection

To test that the CDC_ECM connection is working properly, issue the following command:

```
ip netns exec cdc_ecm-testing ping 8.8.8.8 -c 5
```

If the CDC_ECM session was successful, the response should look something like:


COM85:115200bps - Tera Term VT
File Edit Setup Control Window Help

```
root@linux-pc:~# ip netns exec cdc_ecm-testing ping 8.8.8.8 -c 5
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=52 time=150 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=52 time=153 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=52 time=137 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=52 time=136 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=52 time=135 ms

--- 8.8.8.8 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4002ms
rtt min/avg/max/mdev = 135.530/142.724/153.621/7.784 ms
root@linux-pc:~#
```

4.5.7 Close the CDC_ECM Connection

To stop the CDC_ECM session simply issue the following AT command to the modem:

```
AT#ECMD=0
```

This will turn off CDC_ECM mode on the modem. Next issue the following command to the Linux PC to close the dhclient session:

```
ip netns exec cdc_ecm-testing dhclient -r
```

It is a good idea to close a CDC_ECM session whenever it is no longer needed. This will prevent accidental usage of cellular data.

4.5.8 Delete the Linux Namespace

To delete the Linux namespace, issue the following command:

```
ip netns del cdc_ecm-testing
```

The Linux namespace will also be deleted upon reboot of the Linux PC, so it is not entirely necessary to delete it after testing has concluded.

4.6 CDC_ECM Troubleshooting

If the "`AT#ECM`" command returns "`ERROR`"

1. Verify that the USB composition is set properly on the modem.
    a. Issue "**`AT#USBCFG`**" to the modem.
        i. The modem should respond with "**`#USBCFG: 4`**"
    b. If the modem's USB composition is set incorrectly, issue "**`AT#USBCFG=4`**" to swap to the proper configuration.
        i. The modem will reboot with the correct composition.
2. Verify that the modem does not have an active PDP context.
    a. Issue "**`AT#SGACT?`**" to query active PDP contexts.
        i. If the modem has an active PDP context, close it with one of the following commands:
            1. *Verizon Firmware:*
                a. ***`AT#SGACT=3,0`***
            2. *AT&T Firmware:*
                a. ***`AT#SGACT=1,0`***
    b. After closing any open PDP contexts, issue "**`AT#ECMD=0`**" to ensure that any active CDC_ECM sessions are closed.
    c. Retry the "**`AT#ECM`**" command again.

5. MBIM

5.1 System Set Up

Make sure Section 2 is completed. Specifically, **`libmbim-utils`** needs to be installed and the **`USBCFG`** of the modem should be set to **`2`**.

Finally, make sure the SWDK is plugged into port J15 for USB use. To check, enter the following command:

```
lsusb
```

The output should be along the lines of:

```
Bus 001 Device 008: ID 1bc7:1204 Telit Wireless Solutions
```

Note the **`1bc7`** and **`1204`** from the table earlier. If the cable is plugged into J14, the output will be similar to the following:

```
Bus 001 Device 005: ID 0403:6001 Future Technology Devices
International, LTD FT232 USB-Serial (UART) IC
```

5.2 Using MBIM

5.2.1 Disconnect Other Interfaces

At this point, disconnect from any networks, including wifi and ethernet.

Next, enter the following commands to stop the ModemManager and NetworkManager:

```
service ModemManager stop
service NetworkManager stop
```

These commands don't produce any output to the terminal if successful.

Modem Manager and Network Manager need to be stopped so they don't try to set up the modem automatically while we are trying to. NimbeLink does not support using Modem Manager or Network Manager with the modems.

5.2.2 Gather Information

Now, enter the following command:

```
dmesg | grep cdc
```

The output should be similar to the following:

```
root@probook:/home/kyle# dmesg | grep cdc
[   90.097666] usbcore: registered new interface driver cdc_ncm
[   90.100642] usbcore: registered new interface driver cdc_wdm
[   90.134534] cdc_mbim 3-2:1.2: cdc-wdm0: USB WDM device
[   90.135028] cdc_mbim 3-2:1.2 wwan0: register 'cdc_mbim' at
```

```
usb-0000:00:14.0-2, CDC MBIM, e6:6c:5d:e5:3d:31
[   90.136424] usbcore: registered new interface driver cdc_mbim
```

Take note of the highlighted pieces of information. The yellow highlight denotes a **/dev** file and the blue highlight denotes a network interface. These will be needed later.

Next, enter the following:

```
dmesg | grep wwan
```

Output should be similar to:

```
root@probook:/home/kyle# dmesg | grep wwan
[   90.135028] cdc_mbim 3-2:1.2 wwan0: register 'cdc_mbim' at
usb-0000:00:14.0-2, CDC MBIM, e6:6c:5d:e5:3d:31
[   90.135044] cdc_mbim 3-2:1.2 wwp0s26u1u2i2: renamed from
wwan0
```

The green highlight notes a name change of the network interface from above. This line may not appear. If it does, take note of the name change for the interface, as it will be needed later. The following example assumes that this name change does not occur.

5.2.3 Creating and Configuring MBIM Session

Now we will create and configure the MBIM session. First, issue the following, using the information from above:

```
mbimcli -p -d /dev/cdc-wdm0 --query-subscriber-ready-status
--no-close
```

This command creates a session. The response will be similar to the following:

```
[/dev/cdc-wdm0] Subscriber ready status retrieved:
           Ready state: 'initialized'
         Subscriber ID: '<imei>'
             SIM ICCID: '<iccid>'
            Ready info: 'none'
     Telephone numbers: (1) '++16123xxxxx'
[/dev/cdc-wdm0] Session not closed:
         TRID: '4'
```

Take note of the session that wasn't closed, as it will be used in the following command:

```
mbimcli -p -d /dev/cdc-wdm0 --attach-packet-service --no-close
--no-open=4
```

This command attaches a packet service to the session that was opened from the previous command and leaves the session open. The response should be of the form:

```
[/dev/cdc-wdm0] Successfully attached to packet service
```

```
[/dev/cdc-wdm0] Packet service status:
               Network error: 'unknown'
        Packet service state: 'attached'
      Available data classes: 'lte'
                Uplink speed: '50000000 bps'
              Downlink speed: '100000000 bps'
[/dev/cdc-wdm0] Session not closed:
          TRID: '6'
```

Once again, the session that was not closed will be used in the following command:

```
mbimcli -p -d /dev/cdc-wdm0
--connect=session-id=0,apn=<apn>,ip-type=ipv4 --no-close
--no-open=6
```

This is the final command for setting up the MBIM.

Replace **`<apn>`** with the proper APN for the chosen SIM card and cellular carrier. The proper APN for an AT&T SIM card purchased through NimbeLink is **`iot0718.com.attz`**. The APN for a Verizon SIM card purchased through NimbeLink is **`nimblink.gw12.vzwentp`**

The response from the above command should be similar to the following:

```
[/dev/cdc-wdm0] Successfully connected

[/dev/cdc-wdm0] Connection status:
              Session ID: '0'
        Activation state: 'activated'
        Voice call state: 'none'
                 IP type: 'ipv4'
            Context type: 'internet'
           Network error: 'unknown'

[/dev/cdc-wdm0] IPv4 configuration available: 'address, gateway,
dns, mtu'
          IP [0]: '166.152.x.xxx/xx'
         Gateway: '166.152.x.xxx'
         DNS [0]: '198.224.xxx.xxx'
         DNS [1]: '198.224.xxx.xxx'
             MTU: '1428'

[/dev/cdc-wdm0] IPv6 configuration available: 'none'
[/dev/cdc-wdm0] Session not closed:
          TRID: '9'
```

Once again take note of the highlighted lines, as these will be used to set up the network interface in the following commands.

5.2.4 Activating MBIM

The following commands don't produce responses to the terminal:

```
ip addr add 166.152.x.xxx/xx dev wwan0
ip link set wwan0 mtu 1428 up
ip route add default via 166.152.x.xxx dev wwan0
```

Pay attention to the network interface name (highlighted blue), as this is the name that might have changed earlier.

Once these commands have been entered, the MBIM session should be all set up. Enter the following to ping 8.8.8.8 to test:

```
ping -I wwan0 -c 5 8.8.8.8
```

If the response looks like this, the session has been successfully set up:

```
PING 8.8.8.8 (8.8.8.8) from 166.152.x.xxx wwan0: 56(84) bytes of
data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=51 time=84.8 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=51 time=83.0 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=51 time=89.0 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=51 time=68.2 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=51 time=87.4 ms

--- 8.8.8.8 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 68.233/82.468/88.970/7.410 ms
```

5.2.5 Using DNS

We can also use the Domain Name Services (DNS) from above. Open the **`/etc/resolv.conf`** file with any text editor (for instance, vim), and comment out or delete the existing contents (lines that start with # are commented out). Next, copy and paste the orange highlighted lines from above into the file. The format of the file is:

```
nameserver <ip address>
```

After saving the file, ping google.com with the following command:

```
ping -I wwan0 -c 5 google.com
```

If the response is like the following, everything has been properly set up.

```
PING google.com (172.217.4.46) from 166.152.x.xxx wwan0: 56(84)
bytes of data.
64 bytes from lga15s46-in-f14.1e100.net (172.217.4.46):
icmp_seq=1 ttl=52 time=52.3 ms
64 bytes from lga15s46-in-f14.1e100.net (172.217.4.46):
icmp_seq=2 ttl=52 time=61.2 ms
64 bytes from lga15s46-in-f14.1e100.net (172.217.4.46):
```

```
icmp_seq=3 ttl=52 time=81.9 ms
64 bytes from lga15s46-in-f14.1e100.net (172.217.4.46):
icmp_seq=4 ttl=52 time=78.4 ms
64 bytes from lga15s46-in-f14.1e100.net (172.217.4.46):
icmp_seq=5 ttl=52 time=77.8 ms

--- google.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 52.315/70.329/81.919/11.527 ms
```

5.3 Troubleshooting

If the **`dmesg| grep cdc`** command returns **`cdc_qmi`** instead of **`cdc_mbim`** or the **`dmesg| grep wwan`** command returns nothing, the set up procedure in Section 2 wasn't followed correctly. Stop and revisit the set up procedure to change the modem's **`USBCFG`** and/or **`echo`** the USB VID and PID information into the **`new_id`** file.

If one of the **`ip`** commands is entered wrong, you may not be able to just reenter it correctly. The commands below can help undo what was done.

```
ip link set [interface] down
ip route del default via [ip address] dev [interface]
```

If these commands don't help, resetting the PC will reset the entire command line sequence and give a fresh start.

If all the commands have been entered properly and the **`ping`** command returns with 100% packet loss, check that all the other network interfaces have been brought down (i.e., unplugging the ethernet cable, turning on wifi or entering airplane mode, etc.). Though it is possible to have multiple interfaces up at the same time as MBIM, it is not covered in this guide and additional steps are needed.

Note that **`ping google.com`** won't work unless the DNS addressed have been copied into the **`/etc/resolv.conf`** file. If the two DNS addresses have been added to the file and it still doesn't work, try commenting out the other lines in the file. Lines that start with **`#`** will be ignored.

6. QMI

6.1 System Set Up

Make sure Section 2 is completed. Specifically, **`libqmi-utils`** needs to be installed and the **`USBCFG`** of the modem should be set to **`0`**.

Finally, make sure the SWDK is plugged into port J15 for USB use. To check, enter the following command:

```
lsusb
```

The output should be along the lines of:

```
Bus 001 Device 007: ID 1bc7:1201 Telit Wireless Solutions
```

Note the **`1bc7`** and **`1201`** from the table earlier. If the cable is plugged into J14, the output will be similar to the following:

```
Bus 001 Device 005: ID 0403:6001 Future Technology Devices
International, LTD FT232 USB-Serial (UART) IC
```

6.2 Using QMI

6.2.1 Disconnect Other Interfaces

First, disconnect from any networks, including wifi and ethernet.

Next, enter the following commands to stop the ModemManager and NetworkManager:

```
service ModemManager stop
service NetworkManager stop
```

These commands don't produce any output to the terminal if successful.

Modem Manager and Network Manager need to be stopped so they don't try to set up the modem automatically while we are trying to. NimbeLink does not support using Modem Manager or Network Manager with the modems.

6.2.2 Gather Information

Now, enter the following command:

```
dmesg | grep cdc
```

The output should be similar to the following:

```
root@probook:/home/kyle# dmesg | grep cdc
[   90.097666] usbcore: registered new interface driver cdc_ncm
[   90.100642] usbcore: registered new interface driver cdc_wdm
[   90.134534] cdc_mbim 3-2:1.2: cdc-wdm0: USB WDM device
[   90.135028] cdc_mbim 3-2:1.2 wwan0: register 'cdc_mbim' at
usb-0000:00:14.0-2, CDC MBIM, e6:6c:5d:e5:3d:31
[   90.136424] usbcore: registered new interface driver cdc_mbim
```

Take note of the highlighted pieces of information. The yellow highlight denotes a **`/dev`** file and the blue highlight denotes a network interface. These will be needed later.

Next, enter the following:

```
dmesg | grep wwan
```

Output should be similar to:

```
root@probook:/home/kyle# dmesg | grep wwan
[   90.135028] cdc_mbim 3-2:1.2 wwan0: register 'cdc_mbim' at
usb-0000:00:14.0-2, CDC MBIM, e6:6c:5d:e5:3d:31
[   90.135044] cdc_mbim 3-2:1.2 wwp0s26u1u2i2: renamed from
wwan0
```

The green highlight notes a name change of the network interface from above. This line may not appear. If it does, take note of the name change for the interface, as it will be needed later. The following example assumes that this name change does not occur.

The next few commands are to check the hardware information. First, check the manufacturer:

```
qmicli -d /dev/cdc-wdm0 --dms-get-manufacturer
```

The command should respond with the following:

```
[/dev/cdc-wdm0] Device manufacturer retrieved:
        Manufacturer: 'QUALCOMM INCORPORATED'
```

If it doesn't, QMI is unavailable as it is exclusive to qualcomm chipsets.

The following command confirms that the device is a TC4NAG:

```
qmicli -d /dev/cdc-wdm0 --dms-get-model
```

Response:

```
[/dev/cdc-wdm0] Device model retrieved:
        Model: 'LE910C4-NF'
```

6.2.3 Create Configuration File

Before issuing the command to start the networking, a config file needs to be created. Use any text editor, such as nano or vim, to create the file **`/etc/qmi-network.conf`**. Enter the following:

```
APN=<apn>
PROXY=yes
```

Replace **<apn>** with the proper APN for the chosen SIM card and cellular carrier. The proper APN for an AT&T SIM card purchased through NimbeLink is **iot0718.com.attz**. The APN for a Verizon SIM card purchased through NimbeLink is **nimblink.gw12.vzwentp**.

6.2.4 Start QMI

Once the file has been created and saved, enter the following to start QMI:

```
qmi-network /dev/cdc-wdm0 start
```

The response should be similar to the following:

```
Loading profile at /etc/qmi-network.conf…
         APN: <apn>
         APN user: unset
         APN password: unset
         qmi-proxy: yes
     Checking data format with 'qmicli -d /dev/cdc-wdm0
--wda-get-data-format --device-open-proxy'...
     Device link layer protocol retrieved: raw-ip
     Getting expected data format with 'qmicli -d /dev/cdc-wdm0
--get-expected-data-format'...
     Expected link layer protocol retrieved: 802-3
     Updating kernel link layer protocol with 'qmicli -d
/dev/cdc-wdm0 --set-expected-data-format=raw-ip'...
     Kernel link layer protocol updated
     Starting network with 'qmicli -d /dev/cdc-wdm0
--wds-start-network=apn='<apn>'  --client-no-release-cid
--device-open-proxy'...
     Saving state at /tmp/qmi-network-state-cdc-wdm0... (CID:
17)
     Saving state at /tmp/qmi-network-state-cdc-wdm0... (PDH:
2267733632)
     Network started successfully
```

6.2.5 Configure Networking

Next, get the settings that QMI set up. If the command times out, wait a few seconds before trying a second time.

```
qmicli -d /dev/cdc-wdm0 --wds-get-current-settings
```

The response will be like the following:

```
[/dev/cdc-wdm0] Current settings retrieved:|
           IP Family: IPv4
        IPv4 address: 166.152.x.xxx
    IPv4 subnet mask: 255.255.255.xxx
IPv4 gateway address: 166.152.x.xxx
```

```
       IPv4 primary DNS: 8.8.xxx.xxx
     IPv4 secondary DNS: 8.8.xxx.xxx
                    MTU: 1428
                Domains: none
```

Take note of this output, as the information here will be used in the following commands to set up the networking.

The following three commands don't produce any output to the terminal, but set up the networking. For the **`<converted network mask>`**, see the table following the commands:

```
ip address add <IPv4 address>/<converted network mask> dev wwan0
ip link set wwan0 up
ip route add default via <IPv4 gateway address>
```

The IPv4 subnet mask listed above needs to be converted into slash notation. Check the following table for the conversion:

IPv4 subnet mask	/ notation
255.255.255.255	/32
255.255.255.254	/31
255.255.255.252	/30
255.255.255.248	/29
255.255.255.240	/28
255.255.255.224	/27
255.255.255.192	/26
255.255.255.128	/25
255.255.255.0	/24

After the **`ip`** commands have been set up and the interface is known, the network should be set. Test with the following command:

```
ping -I wwan0 -c 5 8.8.8.8
```

If the response looks like this, the session has been successfully set up:

```
ping 8.8.8.8 (8.8.8.8) from xxx.xxx.x.xxx wwan0: 56(84) bytes of
data.
     64 bytes from 8.8.8.8: icmp_seq=1 ttl=51 time=84.8 ms
     64 bytes from 8.8.8.8: icmp_seq=2 ttl=51 time=83.0 ms
     64 bytes from 8.8.8.8: icmp_seq=3 ttl=51 time=89.0 ms
     64 bytes from 8.8.8.8: icmp_seq=4 ttl=51 time=68.2 ms
     64 bytes from 8.8.8.8: icmp_seq=5 ttl=51 time=87.2 ms

     --- 8.8.8.8 ping statistics ---
     5 packets transmitted, 5 received, 0% packet loss, time
4004ms
     rtt min/avg/max/mdev = 68.233/82.468/88.970/7.410 ms
```

6.2.6 Using DNS

We can also use the Domain Name Services (DNS) from above. Open the `/etc/resolv.conf` file with any text editor (for instance, vim), and comment out or delete the existing contents (lines that start with `#` are commented out). Next, copy and paste the DNS lines from above into the file. The format of the file is:

```
nameserver <ip address>
```

After saving the file, ping google.com with the following command:

```
ping -I wwan0 -c 5 google.com
```

If the response is like the following, everything has been properly set up.

```
ping google.com (172.217.4.46) from 166.152.x.xxx wwan0: 56(84)
bytes of data.
64 bytes from lga15s46-in-f14.1e100.net (172.217.4.46):
icmp_seq=1 ttl=52 time=52.3 ms
64 bytes from lga15s46-in-f14.1e100.net (172.217.4.46):
icmp_seq=2 ttl=52 time=61.2 ms
64 bytes from lga15s46-in-f14.1e100.net (172.217.4.46):
icmp_seq=3 ttl=52 time=81.9 ms
64 bytes from lga15s46-in-f14.1e100.net (172.217.4.46):
icmp_seq=4 ttl=52 time=78.4 ms
64 bytes from lga15s46-in-f14.1e100.net (172.217.4.46):
icmp_seq=5 ttl=52 time=77.8 ms

--- google.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 52.315/70.329/81.919/11.527 ms
```

6.3 Troubleshooting

If the **`qmi`** commands fail, make sure that **`libqmi-utils`** is installed, the set up procedure from Section 2 has been followed, and that the **`\etc\qmi-network.conf`** file is named correctly and contains the correct information.

If the first few lines in the response of **`qmi-network /dev/cdc-wdm0 start`** says **`'Profile not found'`**, make sure the **`/etc/qmi-network.conf`** file is named correctly.